A CONTINUOUS AND A DISCRETE PARTICLE SWARM OPTIMIZATION ALGORITHM FOR UNCAPACITATED FACILITY LOCATION PROBLEM

by

Ali Rıza GÜNER

A thesis submitted to

the Graduate Institute of Sciences and Engineering

of

Fatih University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Industrial Engineering

June 2006
Istanbul, Turkey

APPROVAL PAGE

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____
Prof. Dr. Mazhar ÜNSAL
Head of Department

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____
Assist. Prof. Mehmet ŞEVKLİ
Supervisor

Examining Committee Members

Assist. Prof. Mehmet ŞEVKLİ _____

Assist. Prof. M. Fatih TAŞGETİREN _____

Assist. Prof. Nurullah ARSLAN _____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Sciences and Engineering.

_____
Assist. Prof. Nurullah ARSLAN
Director

Date
June 2006

A CONTINUOUS AND A DISCRETE PARTICLE SWARM OPTIMIZATION ALGORITHM FOR UNCAPACITATED FACILITY LOCATION PROBLEM

Ali Rıza GÜNER

M. S. Thesis - Industrial Engineering
June 2006

Supervisor: Assist. Prof. Mehmet ŞEVKLİ

ABSTRACT

In this paper, a Continuous and a Discrete version of Particle Swarm Optimization (PSO) algorithms proposed for a well known discrete problem, Uncapacitated Facility Location (UFL) problem.

PSO is one of the recent metaheuristics based on evolutionary algorithms invented by Eberhart and Kennedy based on the metaphor of social interaction and communication such as bird flocking and fish schooling. It has been successfully applied to a wide range of applications. On the other hand, developing solution methods for the UFL problem has been a hot topic of research for the last 40 years. Thus PSO algorithms are proposed to solve UFL problems. In order to improve the solution quality local searches are embedded in the PSO algorithms. To make a confidential comparison the proposed PSO algorithms are applied to the benchmark suites collected from OR library. The results are presented and compared with the optimum results in the literature. It is concluded that the proposed PSO algorithms have found optimum results in a reosanable CPU time.

Keywords: Particle Swarm Optimization, Uncapacitated Facility Location Problem, Continuous PSO, Discrete PSO

KAPASİTESİZ TESİS YERİ SEÇİMİ PROBLEMLERİ İÇİN SÜREKLİ VE KESİKLİ PARÇACIK SÜRÜ OPTİMİZASYONU YAKLAŞIMI

Ali Rıza GÜNER

Yüksek Lisans Tezi – Endüstri Mühendisliği
Haziran 2006

Tez Yöneticisi: Yrd. Doç. Dr. Mehmet ŞEVKLİ

ÖZ

Bu tezde bir sürekli ve bir kesikli parçacık sürü optimizasyonu (PSO) algoritması iyi bilinen bir kesikli problem olan kapasitesiz tesis yerleştirme (UFL) problemleri için önerilmiştir.

PSO, Eberhart ve Kennedy tarafından önerilen ve evrimsel algoritmalara dayanan yeni bir sezgisel yöntemdir. Kuş ve balık sürülerinin sosyal iletişimi sırasında oluşan hareketlerinden esinlenilmiş ve şu ana kadar çok çeşitli problemleri çözmek için başvurulmuştur. Diğer taraftan, UFL son kırk yıldır araştırmacılar tarafından çözüm yöntemleri önerilen bir konudur. Bu yüzden UFL problemleri için PSO algoritmaları önerilmiştir. Çözüm kalitesini artırmak için ayrıca bu PSO algoritmalarına birer yerel arama algoritması eklenmiştir. Güvenilir karşılaştırmalar yapabilmek için önerilen PSO algoritmaları yöneylem araştırmaları kütüphanesindeki karşılaştırma problemlerine uygulanmıştır. Sonuçlar ortaya konulmuş ve literatürdeki optimum değerler ile karşılaştırılmıştır. Sonuç olarak önerilen PSO algoritmaları optimum sonuçları kabul edilebilir işlem zamanı içinde elde etmiştir.

Anahtar Kelimeler: Parçacık Sürü Optimizasyonu, Kapasitesiz Tesis Yerleştirme Problemi, Sürekli PSO, Kesikli PSO

To my family,

ACKNOWLEDGEMENT

I would like to thank my thesis supervisor Assist. Prof. Mehmet ŞEVKLİ for his guidance throughout the research.

I express my sincere appreciation to the Assist. Prof. Fatih TAŞGETİREN for his invaluable help during the implementation phase of the thesis.

Thanks go to the jury member Assist. Prof. Nurullah ARSLAN for his valuable suggestions and comments.

I express my thanks and appreciation to my family for their understanding, and motivation. Lastly, but in no sense the least, I also thank my friend Muhammet Balcılar for his encouraging thoughts and comments during my studies.

TABLE OF CONTENTS

ABSTRACT iii
ÖZ iv
DEDICATION v
ACKNOWLEDGMENT vi
TABLE OF CONTENTS vii
LIST OF TABLES ix
LIST OF FIGURES xi
LIST OF SYMBOLS AND ABBREVIATIONS xii
CHAPTER 1 INTRODUCTION 1
CHAPTER 2 UNCAPACITATED FACILITY LOCATION PROBLEM 4
2.1 LITERATURE REVIEW 4
2.2 GENERAL STRUCTURE OF UFL PROBLEM 7
CHAPTER 3 PARTICLE SWARM OPTIMIZATION 9
3.1 INTRODUCTION 9
3.2 LITERATURE REVIEW 12
3.3 THE BASIC ELEMENTS OF PSO 15
3.3.1 Initialization: 15
3.3.2 Particle and Population: 16
3.3.3 Particle velocity: 16
3.3.4 Particle Position: 17
3.3.5 Inertia weight: 18
3.3.6 Fitness Evaluation: 18
3.3.7 Personal best: 18
3.3.8 Global best: 18
3.3.9 Termination criterion: 19
3.4 THE IMPLEMENTATION OF PSO TO A PROBLEM 19
3.4.1 Initialize 19

3.4.2 Update iteration counter....20
3.4.3 Update inertia weight....20
3.4.4 Update velocity....20
3.4.5 Update position....20
3.4.6 Evaluate Fitness....20
3.4.7 Update personal best....20
3.4.8 Update global best....20
3.4.9 Terminate....21
3.5 THE EFFECTS OF THE PARAMETERS....21
3.5.1 Examples of dynamic behavior....21
3.5.2 Effect of the random numbers....23
CHAPTER 4 THE PROPOSED CONTINUOUS AND DISCRETE PSO ALGORITHMS FOR UFL PROBLEM....24
4.1 THE CONTINUOUS PSO ALGORITHM FOR UFL PROBLEM....24
4.2 THE DISCRETE PSO ALGORITHM FOR UFL PROBLEM....28
4.3 THE LOCAL SEARCH ALGORITHM FOR PSO....31
CHAPTER 5 EXPERIMENTAL RESULTS....33
5.1 CONTINUOUS PSO PARAMETERS....34
5.2 DISCRETE PSO PARAMETERS....35
5.3 THE COMPARISON OF CPSO AND DPSO....35
5.3.1 Statistically Testing....37
5.4 THE COMPARISON OF $CPSO_{LS}$ AND $DPSO_{LS}$....39
5.4.1 Statistically Testing....41
CHAPTER 6 CONCLUSIONS....45
REFERENCES....47
APPENDIX A EXPERIMENTAL RESULTS FOR CPSO....52
APPENDIX B EXPERIMENTAL RESULTS FOR DPSO....58
APPENDIX C EXPERIMENTAL RESULTS FOR $CPSO_{LS}$....64
APPENDIX D EXPERIMENTAL RESULTS FOR $DPSO_{LS}$....70
APPENDIX E EXAMPLE OF A RUN'S EXPORTED DATA....76
APPENDIX F EXAMPLE OF A DATASET....77

LIST OF TABLES

TABLE

4.1 An illustration of deriving open facility vector from position vector for a 5-facility problem for CPSO…….….25

4.2 An example of 5-facility to 6-customer problem…….…..26

4.3 An illustration of open facility vector for a 5-facility problem for DPSO…….…30

5.1 Benchmarks tackled with the sizes (number of facilities × number of customers) and the optimum fitness values…….....34

5.2 Experimental Results of CPU and Fitness values for CPSO…….……36

5.3 Experimental Results of CPU and Fitness values for DPSO…….…....37

5.4 Statistic comparison between CPSO and DPSO....…….…….…..38

5.5 Experimental Results of CPU and Fitness values for $CPSO_{LS}$…….….....40

5.6 Experimental Results of CPU and Fitness values for $DPSO_{LS}$…….….41

5.7 Statistic comparison between CPSO and $CPSO_{LS}$……..…...42

5.8 Statistic comparison between DPSO and $DPSO_{LS}$ with t-test.....43

5.9 Statistic comparison between $DPSO_{LS}$ and $CPSO_{LS}$ with t-test……...…..44

A.1 Experimental Results of Fitness and CPU for CPSO (Cap71-74, Cap81-82)…...52

A.2 Experimental Results of Fitness and CPU for CPSO (Cap83-84, Cap91-94)…...53

A.3 Experimental Results of Fitness and CPU for CPSO (Cap101-104, Cap111-112)…….…….....54

A.4 Experimental Results of Fitness and CPU for CPSO (Cap113-114, Cap121-124)……....55

A.5 Experimental Results of Fitness and CPU for CPSO (Cap131-134)…….……56

A.6 Experimental Results of Fitness and CPU for CPSO (CapA, CapB, CapC)…….57

B.1 Experimental Results of Fitness and CPU for DPSO (Cap71-74, Cap81-82)…..58

B.2 Experimental Results of Fitness and CPU for DPSO (Cap83-84, Cap91-94)…..59

B.3 Experimental Results of Fitness and CPU for DPSO (Cap101-104, Cap111-112)……....60
B.4 Experimental Results of Fitness and CPU for DPSO (Cap113-114, Cap121-124)……....61
B.5 Experimental Results of Fitness and CPU for DPSO (Cap131-134)……..…62
B.6 Experimental Results of Fitness and CPU for DPSO (CapA, CapB, CapC)…….63
C.1 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap71-74, Cap81-82)....64
C.2 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap83-84, Cap91-94)....65
C.3 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap101-104, Cap111-112)……....66
C.4 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap113-114, Cap121-124)……....67
C.5 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap131-134)……..68
C.6 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (CapA, CapB, CapC)…..69
D.1 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap71-74, Cap81-82)....70
D.2 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap83-84, Cap91-94)....71
D.3 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap101-104, Cap111-112)……....72
D.4 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap113-114, Cap121-124)…….....73
D.5 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap131-134)……..…74
D.6 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (CapA, CapB, CapC)…..75
F.1 Structure of a Dataset….....77
F.2 Dataset of problem Cap71 (a)……..……..….78
F.3 Dataset of problem Cap71 (b)……..……..….79

LIST OF FIGURES

FIGURE

3.1 Initializing example for the ith particle……..16
3.2 The pseudo code of PSO algorithm……...21
3.3 Examples of dynamic behavior of a single particle for various choices of the parameters a and b……..22
(a) Harmonic oscillations with slow convergence.
(b) Harmonic oscillations with quick convergence.
(c) Harmonic oscillations with zigzagging.
(d) Non-oscillatory convergence.
(e) Symmetric zigzagging.
(f) Asymmetric zigzagging.
4.1 Pseudo code of CPSO algorithm for UFL problem……...27
4.2 Pseudo code of DPSO algorithm for UFL problem ……..30
4.3 Pseudo code for local search……..32
E.1 Example of a run's exported data ……..76

LIST OF SYSMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

PSO : Particle Swarm Optimization Algorithm

UFL : Uncapacitated Facility Location

DPSO : Discrete Particle Swarm Optimization Algorithm

CPSO : Continuous Particle Swarm Optimization Algorithm

$DPSO_{LS}$: Discrete PSO with Local Search

$CPSO_{LS}$: Continuous PSO with Local Search

CPU : Central Processing Unit

NP-hard : Non-polinomial hard

x_{ij} : the quantity supplied from facility *i* to customer *j*;

y_i : indicates whether facility j is established ($y_i = 1$) or not ($y_i = 0$).

Y_i : represents the opening or closing facilities identified based on the position vector(X_i)

y_{ik} : represents opening or closing the k^{th} facility of the i^{th} particle.

n : number of sites

m : number of customers,

fc_j : fixed cost of a site

c_{ij} : transport cost from each site to each customer

X_i^t : i^{th} particle in the swarm at iteration t; $X_i^t = \left[x_{i1}^t, x_{i2}^t, .., x_{in}^t\right]$

x_{ij}^t : Position value of the ith particle with respect to the jth dimension ($j = 1,2,...,n$).

X_{max} : maximum position

X_{min} : minimum position

pop^t : Set of ρ particles in the swarm at iteration t, i.e., $pop^t = \left[X_1^t, X_2^t, ..., X_\rho^t\right]$

V_i^t : Velocity of particle i at iteration t; $V_i^t = \left[v_{i1}^t, v_{i2}^t, ..., v_{in}^t\right]$

v_{ij}^t : Velocity of particle i at iteration t with respect to the j[th] dimension.

V_{max} : maximum velocity

V_{min} : minimum velocity

w^t : Inertia weight; a parameter to control the impact of the previous velocities on the current velocity.

P_i^t : The best position of the particle i with the best fitness until iteration t, personal best; $P_i^t = \left[p_{i1}^t, p_{i2}^t, ..., p_{in}^t\right]$

p_{ij}^t : Position value of the i[th] personal best with respect to the j[th] dimension ($j = 1,2,...,n$).

G^t : The best position of the globally best particle achieved so far, global best; $G^t = \left[g_1^t, g_2^t, ..., g_n^t\right]$

g_j^t : Position value of the global best with respect to the j[th] dimension ($j = 1,2,...,n$)

c_1 : cognition learning rate

c_2 : social learning rate

r_1 , r_2 : random constant numbers

ρ : a uniformly generated number between 0 and 1

η and κ : randomly generated parameters

K : constriction factor

s_0 : the final produced solution

$ARPE$: the percentage of difference from the optimum

H_i : the i^{th} replication solution value

U : the optimal value provided in the literature

R : the number of replications

HR : the ratio between the number of runs yielded the optimum and the total numbers of experimental trials.

CHAPTER 1

INTRODUCTION

Partly as a consequence of globalization, the selection of a facility's location has become extremely complicated. Furthermore, selection of a suboptimal location could be very costly. Location problems are one of the most widely studied problems in NP-hard (Cornuéjols et al., 1990) combinatorial optimization problems thus there is a very rich literature in operations research (*OR*) for this kind of problem (Mirchandani and Francis, 1990). Based on some assumptions, location problems can be classified into four basic categories: p-median problems, p-center problems, uncapacitated facility location problems, and capacitated facility location problems (Ghosh, 2002).

In the uncapacitated facility location (*UFL*) problem the cost of satisfying the client requirements has two components a fixed cost component of setting up a facility in a given site, and a transportation cost component of satisfying the customer requirements. Capacities of all the facilities in the *UFL* problems are assumed to be infinite (Ghosh, 2002).

Performance of a solution method is determined through the execution time of the algorithm and the quality of the results. In order to be qualified, a method should give logical outputs and should achieve the objective of the study optimally.

UFL problems can be solved by mathematical (exact) models such as dynamic programming, branch & bound or integer linear programming algorithms. If the complexity of the problem is a little bit increased, it is still possible to solve the problem polynomially by making some assumptions.

Exact methods are able to solve only small sized problems. As a matter of fact exact algorithms are useful when the number of sites and customers are less. If we attempt to solve complex problems with exact methods, even we are sure that we will get the optimal results at last, our lives may not allow completing the runs; since enumerating the problem takes very long times. That enforced researchers to use heuristics and metaheuristics methods that each has good reputation in solving various combinatorial and real world problems.

Heuristics reveal invaluable solutions. Known heuristic methods start with a single solution and try to develop better solutions in the next generations from the solution currently at hand. Worse solutions are not accepted. Therefore, the execution terminates at the first local minimum being trapped. Thus metaheuristics methods that not permit to trap in the first local optima have been introduced.

Suppose the following scenario: a group of birds are randomly searching for food in an area. There is only one piece of food in the area being searched. All of birds do not know where the food is. A large number of birds synchronously, change direction suddenly, and scatter and regroup together. Each individual benefits from the experience of its own and that of the other members of the swarm during the search for food. But they know how far the food is in each iteration. So what is the best strategy to find the food? The effective one is to follow the bird that is nearest to the food.

Inspired by the scenario and used it to solve the optimization problems, Kennedy and Eberhart (1995) presented the Particle Swarm Optimization. In *PSO*, each single solution is a *bird* in the search space. We call it *particle*. Since *PSO* is population-based and evolutionary in nature, the members in a *PSO* algorithm tend to follow the leader of the group, i.e., the one with the best performance. The particles move around in a multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. All the particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

The organization of this thesis is as follows. Chapter 2 introduces the uncapacitated facility location problem and literature review about it. With which

techniques UFL problems have been tried to solve. Chapter 3 presents the particle swarm optimization, benchmark with some other metaheuristics and for which problems PSO has been proposed. In Chapter 4 the proposed *Continuous PSO (CPSO)* and *Discrete PSO (DPSO)* algorithms for UFL problems and implementing of *Local Search* algorithm to both *CPSO* and *DPSO* is presented. In Chapter 5 the proposed algorithms experimental results are given. Chapter 6 is the Conclusions of the study.

CHAPTER 2

UNCAPACITATED FACILITY LOCATION PROBLEM

The success or failure of business and public facilities depends in a large part on their locations. Effective supply chain management has led to increased profit, increased market share, reduced operating cost, and improved customer satisfaction for many businesses. One strategic decision in supply chain management is facility location (Simchi-Levi et al., 2000). Due to their strategic nature, facility location problems have been widely studied by researchers and practitioners over many years.

In this chapter, literature review about *UFL* problem and general structure of uncapacitated facility location problem will be presented.

2.1 LITERATURE REVIEW

There are different titles for the uncapacitated facility location problem in the literature: the problem of a non recoverable tools optimal system (Beresnev et al., 1978); the standardization and unification problem (Gimady, 1970); the optimal parameter problem for the uniform technical system (Beresnev, 1971); the location of bank accounts problem (Cornuejols et al., 1977); warehouse location problem (Khumawala, 1972); uncapacitated facility location problem (Krarup and Pruzan, 1983) etc. Because of the academic interest to one investigation this problem has different interpretations as for many mathematical models. The terminology of Uncapacitated Facility Location (*UFL*) problem to describe the model is used although the others interpretations will possible too.

Developing solution methods for the *UFL* problem has been a hot topic of research for the last 40 years. Thus, *UFL* problems have been studied and examined extensively (Kratica et al., 2001) by various attempts and approaches. Because the *UFL* problem is NP-hard (Cornuéjols, 1990) exact algorithms may not be able to solve large practical problems. Since they are NP-Hard problems, the larger the size of the problem, the harder to find the optimal solution and furthermore, the longer to reach reasonable results. All important approaches relevant to *UFL* problems can be classified into two main categories: exact algorithm such as branch and bound, primal and dual ascent methods, linear programming and Lagrangean relaxation algorithms and metaheuristic based methods (Aydin and Fogarty, 2004). Krarup and Pruzan (1983) and Cornuéjols et al. (1990) gave excellent surveys and reviews of applications and solution methods.

There are a variety of exact algorithms for the *UFL* problem, such as the dual approach of Erlenkotter (1978) and the primal-dual approaches of Körkel (1989). Erlenkotter (1978) developed a dual approach *(DUALOC)* for the *UFL* problem. The *DUALOC* algorithm (Erlenkotter, 1978) is one of the most respected methods based on OR approaches as the fastest one for *UFL* problems for a long time. It is based on a linear programming dual formation *(LP dual)* in condensed form that evolved in simple ascent and adjustment procedures. If ascent and adjustment procedures do not find the optimal solution, Branch and- Bound *(BnB)* procedure completes the solution process. Although this dual approach is an exact algorithm, it can also be used as a heuristic to find good solutions.

Guignard (1985) proposed to strengthen the separable Lagrangean relaxation of the *UFL* problems by using Bender's inequalities generated during a Lagrangean dual ascent procedure. The coupling of that technique with a good primal heuristic could reduce the integrity gap. Simao and Thizy (1989) presented a streamlined dual simplex algorithm designed on the basis of a covering formulation of the *UFL* problem. Their computational experience with standard data sets indicates the superiority of dual approaches. Körkel (1989) showed how to modify a primal-dual version of Erlenkotter's (1978) exact algorithm to get an improved procedure. The computational experience with large-scale problem instances indicated that speedup to *DUALOC* is significant (more than one order of magnitude). Conn and Cornuéjols (1990) present a method based upon the exact solution of the condensed dual of *LP* relaxation via

orthogonal projections. In Holmberg (1995) and Holmberg and Jornsten (1996) a primal-dual solution approach based on decomposition principles is used. They fixed some variables in the primal sub-problem and relaxed some constraints in the dual sub-problem. By fixing their Lagrange multipliers, both of these problems become easier to solve than the original one. The computational tests proved the advantageous in comparison to the dual ascent method of Erlenkotter.

Kratica et al. (2001) have applied genetic algorithms to *UFL* problems to solve 1000x1000-sized customer-facility instances. They considered many benchmarks within the literature to be solved by their algorithm in addition to their own similar large size problems. They compared their results with *DUALOC* algorithm showing that their algorithm is much more efficient than *DUALOC* for problems larger than 100x1000. Although *DUALOC* has better results for some benchmarks, it is worse in time consumption.

Kuehn and Hamburger (1963) developed the first heuristic that has two phases. The first phase is a greedy approach, called the *ADD* method that starts with all facilities closed, keeps adding (opening) the facility resulting in the maximum decrease in the total cost (1), and stops if adding any more facility will no longer reduce the total cost. The second phase is a local search method in which an open facility and a closed facility are interchanged as long as such an interchange reduces the total cost. Another greedy heuristic is the *DROP* method that starts with all facilities open, keeps dropping (closing) the facility that gives the maximum decrease in the total cost, and stops if dropping any more facility will no longer reduce the total cost (Daskin, 1995). These early heuristics provided the basis for many sophisticated heuristics and provided an initial incumbent for many exact solution algorithms. Al-Sultan and Al-Fawzan (1999) presented a tabu search algorithm which produce very good solutions but takes significant computing time and limits the applicability of the algorithm. Michel and Van Hentenryck (2004) proposed tabu search and generated very robust solutions. Sun (2005) presents another tabu search procedure which tested against the Lagrangian method and heuristic procedures reported by Ghosh (2002) and Resende and Werneck (2003). In addition artificial neural network approaches have been proposed to solve *UFL* problems by Gen et al. (1996) and Vaithyanathan et al. (1996).

Jaramillo et al. (2002) applied a genetic algorithm that is mainly based on the operators that Beasley and Chu (1996) applied in their genetic algorithms for covering problems. They compared their results with a Lagrangean relaxation algorithm presented by Beasley (1993). Although Kratica et al. (2001) have proposed a very similar approach, Jaramillo et al. (2002) give a fresher and less time consuming approach. Alves and Almeida (1992) presented simulated annealing algorithms which produce high quality solutions but are quite expensive in computation times. Aydin and Fogarty (2004) presented a distributed evolutionary simulated annealing algorithm implementation that can get the better quality of solutions within shorter times.

In addition, many successful *UFL* model applications have been provided to some problems. The bank account location problem, network design, vehicle routing, distributed data and communication networks (Ghosh,2002), computer network design, cluster analysis, machine scheduling, economic lot sizing, portfolio management (Gen et al., 1996) are some instances without facilities to locate problems that can be modeled as an *UFL* problem.

2.2 GENERAL STRUCTURE OF UFL PROBLEM

There are a variety of models representing a variety of facility location problems. Most of these problems are combinatorial in nature. Based on some assumptions, location problems can be classified into four basic categories:

- Uncapacitated Facility Location (*UFL*)

No limits for capacity, no idea on the number of sites to open

- Capacitated Facility Location (*CFL*)

There are limits for capacities, no idea on the number of sites to open

- p-Median Problems (*p-MP*)

Fixed number of open sites, evaluating by *minisum*

- p-Centre Problems (*p-CP*)

Fixed number of open sites, evaluating by *minimax*

In a *UFL* problem, a fixed cost is associated with the establishment of each facility and a fixed cost is associated with the opening and using of each road from a

customer to a facility. The objective of a *UFL* problem is to decide where to locate the facilities and which roads to use so as to minimize the total cost. There are a number of sites, n and a number of customers, m. Each site has a fixed cost fc_i. There is a transport cost from each site to each customer c_{ij}. There is no limit of capacity for any candidate site and the whole demand of each customer has to be assigned to one site. We are asked to find the number of sites (facilities) to be established and specify those sites such that the total cost will be minimized (2.1). The mathematical formulation of *UFL* can be stated as follows (Cornuéjols et al., 1990):

$$Z = \min\left(\sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij} \cdot x_{ij} + \sum_{i=1}^{n} fc_i \cdot y_i \right) \tag{2.1}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = 1 \tag{2.2}$$

$$0 \leq x_{ij} \leq y_i \text{ and } y_i \in \{0; 1\}; \tag{2.3}$$

where:

$i = 1,\ldots,n$; $j = 1,\ldots,m$

x_{ij} represents the quantity supplied from facility i to customer j;

y_i indicates whether facility i is established ($y_i = 1$) or not ($y_i = 0$).

The constraint (2.2) makes sure that all demands have been met by the open sites, and the constraint (2.3) is to keep integrity. Since it is assumed that there is no capacity limit for any facility, the demand size of each customer is ignored, and therefore constraint (2.2) established without considering demand variable.

CHAPTER 3

PARTICLE SWARM OPTIMIZATION

3.1 INTRODUCTION

Exact methods are able to solve only small sized problems. As a matter of fact exact algorithms are useful when the number of sites and customers are less. If we attempt to solve complex problems with exact methods, even we are sure that we will get the optimal results at last, our lives may not allow completing the runs; since enumerating the problem takes very long times.

Heuristics reveal invaluable solutions. Known heuristic methods start with a single solution and try to develop better solutions in the next generations from the solution currently at hand. Worse solutions are not accepted. Therefore, the execution terminates at the first local minimum being trapped.

Note that, a landscape of an objective function does not have a continual increase or decrease. Namely, let's say in a minimization problem, two succeeding minima may have a maximum point in between. If our solution is closer to the local minimum, then we have the risk to be trapped at this local minimum. Not accepting the worse solutions will hinder to overcome the hill and reach the global minimum. Thus, we can say that, heuristics do not always offer us optimal solutions.

Heuristics were developed for the minimizing the number of problems or they were used for initializing the solution or the solution set of metaheuristics.

Recently metaheuristics are of the greatest interest; since they give the optimal or near-optimal solutions in a shorter time than exact algorithms do. Mostly used

metaheuristics are the Simulated Annealing, Tabu Search, Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization etc.

In Simulated Annealing and Tabu Search, solutions are obtained from an initially formed solution; whereas in Genetic Algorithms, Particle Swarm Optimization *(PSO)* or Ant Colony Optimization methods, the solutions are obtained from an initially constructed population.

The solutions obtained from the metaheuristics methods can be improved by hybridizing the algorithm with local search. The algorithm starts with a complete solution and tries to find a better solution by using the neighborhood of the current solution. We call the solutions as neighbors if the latter solution can be obtained by modifying the current one.

Metaheuristics are able to solve difficult problems in few minutes. Since in metaheuristics, worse solutions are given an opportunity, being trapped at local optima is prevented. Therefore, the solution quality will be increased if metaheuristic methods are used.

In *PSO,* instead of using more traditional genetic operators, each particle (individual) adjusts its "flying" according to its own flying experience and its companions' flying experience. On one hand, it can be counted as an evolutionary method with its way of exploration via neighborhood of solutions (particles) across a population (swarm) and exploiting the generational information gained. On the other hand, it is different from other evolutionary methods in such a way that it has no evolutionary operators such as crossover and mutation. Another advantage is its ease of use with fewer parameters to adjust. In *PSO*, the potential solutions, so-called particles, move around in a multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm.

PSO is distinctly different from other evolutionary-type methods in a way that it does not use the filtering operation (such as crossover and/or mutation) and the members of the entire population are maintained through the search procedure so that

information is socially shared among individuals to direct the search towards the best position in the search space.

In *PSO*, the system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike *GA, PSO* has no evolution operators such as crossover and mutation. In *PSO*, the potential solutions, called particles, fly through the problem space by following the current optimum particles. Compared to *GA*, the advantages of *PSO* are that it is easy to implement and there are fewer parameters to adjust (Allahverdi and Al-Anzi, 2006).

There are also some studies that compare the *PSO* with some other heuristic techniques. To illustrate, Allahverdi, and Al-Anzi (2006) compared a tabu search and a *PSO* for the assembly scheduling Problem and they asserted *PSO* heuristic performs very well for difficult problems. Salman et al. (2003) showed that the *PSO* algorithm solution quality is better than that of GA in most of the test cases for Task Assignment Problem. Moreover, the *PSO* algorithm runs faster as compared with *GA*.

Nature-inspired algorithms are useful because they are based upon well-known models. The underlying physics of such models can act as a guide on how to structure algorithms, and may inspire the confidence to try them over other types of algorithms. Random selection, self-organization, distributed computation, and emergent, shall we say ‘swarm’, intelligence are all attractive features of these algorithms.

Suppose the following scenario: a group of birds are randomly searching for food in an area. There is only one piece of food in the area being searched. All of birds do not know where the food is. A large number of birds synchronously, change direction suddenly, and scatter and regroup together. Each individual benefits from the experience of its own and that of the other members of the swarm during the search for food. But they know how far the food is in each iteration. So what is the best strategy to find the food? The effective one is to follow the bird that is nearest to the food.

Inspired by the scenario and used it to solve the optimization problems, Kennedy and Eberhart (1995) presented the Particle Swarm Optimization in 1995. In *PSO*, each single solution is a “bird” in the search space. We call it “particle”. Since *PSO* is population-based and evolutionary in nature, the members in a *PSO* algorithm tend to

follow the leader of the group, i.e., the one with the best performance. The particles move around in a multi-dimensional search space with a velocity, which is constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. All the particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

There are four *PSO* models defined by Kennedy. The complete velocity update formula is named as the *Full Model.* If the cognition component, c_1 is omitted, it is defined as the *Social-Only Model* and if social component, c_2 is omitted, then the model is called the *Cognition-Only Model.* And the fourth model is the *Selfless Model* which is a kind of *Social-Only Model.* In this model, it selects its global best only from its neighbors.

In this chapter, literature review, the basic elements of *PSO*, the implementation of *PSO* to a problem and the effects of the parameters on *PSO* will be presented respectively.

3.2 LITERATURE REVIEW

The particle swarm concept originated as a simulation of a simplified social system. The original intent was to graphically simulate the graceful but unpredictable choreography of a bird flock. Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variables, and incorporate multidimensional search and acceleration by distance (Kennedy and Eberhart, 1995, Eberhart and Kennedy, 1995). At some point in the evolution of the algorithm, it was realized that the conceptual model was, in fact, an optimizer. Through a process of trial and error, a number of parameters extraneous to optimization were eliminated from the algorithm, resulting in the very simple original implementation (Eberhart et al., 1996).

Two variants of the *PSO* algorithm are developed, namely *PSO* with a local neighborhood, and *PSO* with a global neighborhood. According to the global neighborhood, each particle moves towards its best previous position and towards the

best particle in the whole swarm, called gbest model. On the other hand, according to the local variant so called p_{best}, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood (Kennedy and Eberhart, 2001). It has some tuning parameters which influence the performance of the algorithm; the exploration and exploitation tradeoff. In the work of Eberhart et al. (1996), it was realized that some of the parameters were redundant, and they removed from the original algorithm. The mathematical equations of the original version of *PSO* is as,

$$v_{ij}^{t+1} = v_{ij}^{t} + c_1.r_1.\left(p_{ij}^{t} - x_{ij}^{t}\right) + c_2.r_2.\left(g_j^{t} - x_{ij}^{t}\right) \tag{3.1}$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \tag{3.2}$$

c_1: cognition learning rate

c_2: social learning rate

r_1 and r_2: random constant numbers

Trelea (2003) gives some insights about parameter selection in *PSO*. According to the article, some parameters can be discarded; since they add no value to the algorithm. Trelea (2003) analyzes the deterministic *PSO* algorithm for its dynamic behavior and convergence property.

The velocities of particles' on each dimension (v_{ij}) are restricted to V_{max}. A larger V_{max} facilitates global exploration, while smaller V_{max} facilitates local exploitation. Shi and Eberhart (1998 a and b) added the inertia weight as a constant to the velocity in order to control the exploration and exploitation. The use of inertia weight improved the performance of the algorithm in many applications.

$$v_{ij}^{t+1} = w.v_{ij}^{t} + c_1.r_1.\left(p_{ij}^{t} - x_{ij}^{t}\right) + c_2.r_2.\left(g_j^{t} - x_{ij}^{t}\right) \tag{3.3}$$

w: inertia weight

Clerc (1999) introduced the constriction factor (κ) to *PSO*. It controls, constrains velocities and thus insures convergence. κ negated the need for V_{max}.

$$v_{ij}^{t+1} = \kappa.\left\lfloor w.v_{ij}^{t} + c_1.r_1.\left(p_{ij}^{t} - x_{ij}^{t}\right) + c_2.r_2.\left(g_j^{t} - x_{ij}^{t}\right)\right\rfloor \tag{3.4}$$

$$\kappa = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \text{ where } \varphi = c_1 + c_2, \quad \varphi > 4, \text{ so } \kappa \cong 0.729 \tag{3.5}$$

Eberhart and Shi (2001b) demonstrated that although previous evolutionary paradigms can generally solve static problems, *PSO* can successfully optimize dynamic systems. It can not be known when a larger or a smaller inertia weight is needed. Therefore, that value is set to a dynamic value which starts from 0.9 and descends linearly till 0.4. When it reaches to 0.4 it is increased again to 0.9.

Six years after the introduction of *PSO* Eberhart and Shi reviewed the development, applications and the written books and articles (Eberhart and Shi, 2001a)

Although the applications of *PSO* on combinatorial optimization problems are still limited, *PSO* has its merit in the having a simple concept, low computational cost, cognitive memory, maintaining a population of solution, and having an elegant productive cooperation between its populations (Salman, 2003). After *PSO* was first introduced by Eberhart and Kennedy (1995) and Kennedy and Eberhart (1995), it has been successfully applied to optimize various continuous nonlinear functions. It was applied successfully since then to a handful of computer science and engineering problems (Kennedy and Eberhart, 1995), (Kennedy et al., 2001), (Shi and Eberhart, 1999), (Sugantha, 1999), (Ozcan and Mohan, 1999), (Clerc, 1999). Some of the wide application areas of *PSO* are power and voltage control (Yoshida, 2000), neural network training (Van den Bergh et al., 2000) task assignment (Salman et al., 2003), scheduling problems (Allahverdi and Al-Anzi, 2006), mass-spring system (Brandstatter and Baumgartner, 2002), supplier selection and ordering problem (Yeh, 2003), and other areas where GA can be applied (Allahverdi and Al-Anzi, 2006). More literature can be found in reference (Kennedy and Eberhart, 2001).

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms *(GA)* (Eberhart and Shi,1998). Applications, parameter selection,

and a modified version of *PSO* have been considered, respectively, by Eberhart and Shi (2001) and Shi and Eberhart (1998a), (1998b).

The *PSO* models the social dynamics of flocks of birds and serves as an optimizer for both of continuous and discrete functions. The convergence and parameterization aspects of the *PSO* have been discussed thoroughly (Parsopoulos and Vrahatis,2002), (Clerc and Kennedy,2002), (Trelea, 2003).

It has also been shown that a hybrid strategy which embeds a local optimizer such as hill-climbing in between the iterations of a metaheuristic algorithm can improve the performance significantly (Michalewicz and Fogel, 2002).

Since *PSO* is developed for continuous optimization problem initially, most existing *PSO* applications are resorting to continuous function value optimization (Eberhart and Shi, 1998; Kennedy and Eberhart, 1995). Recently a few researches have been conducted for discrete combinatorial optimization problems. Kennedy and Eberhart (1997) introduced a discrete binary version of *PSO* in 1997. Salman et al. (2003) apply *PSO* for solving task assignment problem. Yin (2004) proposed a discrete PSO algorithm for optimal polygonal approximation of digital curves. Liao (in press) et al. presented another discrete algorithm for flowshop scheduling problems and lastly Pan et al. proposed another for solving the no-wait flowshop scheduling problems.

3.3 THE BASIC ELEMENTS OF PSO

In a *PSO* algorithm, each member is called “particle”, and each particle flies around in the multi-dimensional search space with a velocity, which is constantly updated by the particle’s own experience and the experience of the particle’s neighbors or the experience of the whole swarm. The basic elements of the *PSO* algorithm can be summarized as follows

3.3.1 Initialization:

In *PSO*, each particle corresponds to a candidate solution of the underlying problem. Thus, let each particle represent a decision to solve problem using a vector of

m elements, and each element is an integer value between 1 to *r*. Fig. 3.1 shows an illustrative example for the i^{th} particle which corresponds to a task assignment that assigns five tasks to three processors, and particle *i*, 5=2 means that task 5 is assigned to processor 2. The *PSO* randomly generates an initial swarm of *n* particles, where *n* is the swarm size. These particle vectors will be iteratively modified at each iteration, *t* based on collective experiences in order to improve their solution quality (YinT et al., 2005).

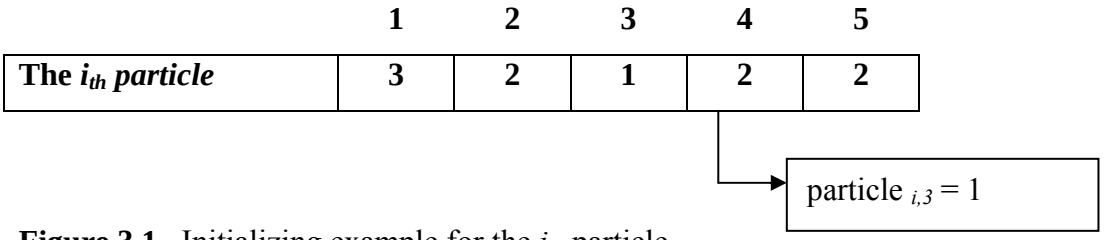

Figure 3.1 Initializing example for the i_{th} particle

3.3.2 Particle and Population:

In *PSO*, each single solution is a "bird" in the search space. We call it *"particle"*. All the particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. X_i^t denotes the i^{th} particle in the swarm at iteration *t* and is represented by *n* number of dimensions as $X_i^t = \left[x_{i1}^t, x_{i2}^t, .., x_{in}^t\right]$, where x_{ij}^t is the position value of the i^{th} particle with respect to the j^{th} dimension ($j = 1,2,...,n$).

Population; pop^t is the set of ρ particles in the swarm at iteration *t*, i.e., $pop^t = \left[X_1^t, X_2^t, ..., X_\rho^t\right]$.

3.3.3 Particle velocity:

To simulate the bird flocking for food foraging, the particle vectors are iteratively modified during the *PSO* evolution. According to the fitness values of these particle vectors, each particle remembers the best vector it experienced so far, referred to as P_i, and the best vector experienced by its neighbors, *G*. The particle's neighbors are defined as the particles within its topological neighborhood in the solution space. There are two versions for keeping the neighbors' best vector, namely p_{best} and g_{best}. In the local version, each particle keeps track of the best vector p_{best} attained by its local topological

neighborhood of particles. In many applications, the neighborhood size is set to about 15% of the swarm size. For the global version, the best vector g_{best} is determined by any particles in the entire swarm. Hence, the g_{best} model is a special case of the p_{best} model (YinT et al., 2005).

During each PSO iteration, particle *i* adjusts its velocity v_{ij} and position vector x_{ij} through each dimension *j* by referring to, with random multipliers, the personal best vector *(P_i)* and the swarm's best vector (*G,* if the global version is adopted) using equations (3.2) and (3.3).

V_i^t is the velocity of particle *i* at iteration *t*. It can be defined as $V_i^t = \left[v_{i1}^t, v_{i2}^t, ..., v_{in}^t\right]$, where v_{ij}^t is the velocity of particle *i* at iteration *t* with respect to the j^{th} dimension.

The particle's velocity on each dimension is set restricted by a maximum velocity v_{max}, which controls the maximum travel speed during each iteration to avoid this particle flying past good solutions. If a velocity on a dimension of a particle exceeds V_{max}, then it is limited to V_{max}. V_{max} controls the exploration and exploitation ability of a particle. It helps to search the regions between the current position and the target position.

Fine-tuning V_{max} is so important that a large value of V_{max} facilitates global exploration, while a smaller V_{max} encourages local exploitation. If V_{max} is set too high or too small, the particles can't explore the search space sufficiently and they could stuck at local optima.

3.3.4 Particle Position:

X_i^t is the position of particle *i* at iteration *t*. It can be defined as $X_i^t = \left[x_{i1}^t, x_{i2}^t, ..., x_{in}^t\right]$, where x_{ij}^t is the position of particle *i* at iteration *t* with respect to the j^{th} dimension.

The particle's position on each dimension is set restricted by a maximum position x_{max}, which controls the maximum travel distance during each iteration to avoid this particle flying past good solutions. If a position on a dimension of a particle exceeds

X_{max}, then it is limited to X_{max}. X_{max} controls the exploration and exploitation ability of a particle. It helps to search the regions between the current position and the target position. It is usually set to 4 or 5 times of V_{max}.

3.3.5 Inertia weight:

w^t is a parameter to control the impact of the previous velocities on the current velocity. When suitably set, the inertia weight helps to balance the local and global exploration, thus the optimal value can be obtained in a few iterations. High values encourage global exploration, while low values facilitate local exploitation.

3.3.6 Fitness Evaluation:

The objective function of the problem can be used to measure the quality of each particle vector. However, this value is discredited if the particle vector violates at least one of the problem constraints. In modern heuristics, infeasible solutions also provide valuable clue to targeting the optimal solution (Michalewicz and Fogel, 2002). The degree of infeasibility for trial solutions can be measured and transformed to a penalty which grows in proportional to the infeasibility level, thus guiding the search toward feasible space (YinT et al., 2005).

3.3.7 Personal best:

P_i^t represents the best position of the particle *i* with the best fitness until iteration *t*, so the best position associated with the best fitness value of the particle *i* obtained so far is called the personal best. For each particle in the swarm, P_i^t can be determined and updated at each iteration *t*. For each particle, the personal best is defined as $P_i^t = \left[p_{i1}^t, p_{i2}^t, ..., p_{in}^t\right]$ where p_{ij}^t is the position value of the i^{th} personal best with respect to the j^{th} dimension ($j = 1,2,...,n$).

3.3.8 Global best:

G^t denotes the best position of the globally best particle achieved so far in the whole swarm. The global best is then defined as $G^t = \left[g_1^t, g_2^t, ..., g_n^t\right]$ where g_j^t is the position value of the global best with respect to the j^{th} dimension ($j = 1,2,...,n$).

3.3.9 Termination criterion:

It is a condition that the search process will be terminated. It might be a maximum number of iteration or maximum CPU time to terminate the search.

3.4 THE IMPLEMENTATION OF PSO TO A PROBLEM

The complete computational procedure of a *PSO* algorithm to a problem can be summarized as follows:

3.4.1 Initialize

The first step of the *PSO* is, like other algorithms, the initializing of the elements. The initializing procedure is usually as follows:

- Set n=0, m=equal (optional) the number of dimensions.
- Generate n particles randomly as explained before, $\{X_i^0, i = 1,..,m\}$ where $X_i^0 = [x_{i1}^0,..,x_{in}^0]$.
- Generate initial velocities of particles randomly $\{V_i^0, i = 1,..,m\}$ where $V_i^0 = [v_{i1}^0,..,v_{in}^0]$
- Evaluate each particle i in the swarm using the objective function f_i^0 for $i = 1,..,m$.
- For each particle i in the swarm, set $P_i^0 = X_i^0$, where $P_i^0 = [p_{i1}^0 = x_{i1}^0,.., p_{in}^0 = x_{in}^0]$ along with its best fitness value, $f_i^p = f_i^0$ for $i = 1,..,m$.
- Find the best fitness value $f_i^0 = \min\{f_i^0\}$ for $i = 1,..,m$ with its corresponding position X_l^0 .
- Set global best to $G^0 = X_l^0$ where $G^0 = [g_1 = x_{l,1},.., g_n = x_{l,n}]$ with its fitness value $f^g = f_l^0$

3.4.2 Update iteration counter

If one of the pre-determined termination criterions have not met update iteration counter: $t = t + 1$

3.4.3 Update inertia weight

If dynamic inertia weight is employed: $w^t = w^{t-1} * \alpha$ where α is decrement factor and usually taken 0,975.

3.4.4 Update velocity

After finding the two best values (P_i and G), the particle updates its velocity with the equation (3.3)

3.4.5 Update position

After finding the new velocity *(V_i),* the particle updates its position vector with the equation (3.2)

3.4.6 Evaluate Fitness

Each particle vector in the swarm is assigned a fitness value indicating the merit of this particle vector such that the swarm evolution is navigated by best particles. The higher the fitness value is, the better the quality of the particle vector is for maximization problems and vice versa for minimization problems.

3.4.7 Update personal best

Each particle is evaluated by using fitness function to see if personal best will improve. That is, if $f_i^t < f_i^p$ for $i = 1,..,m$, then personal best is updated as $P_i^t = X_i^t$ and $f_i^p = f_i^t$ for $i = 1,..,m$.

3.4.8 Update global best

Find the minimum value of personal best. $f_i^t = \min\{f_i^p\} for \quad i = 1,..,m$. If $f_l^t < f^g$, then the global best is updated as $G^t = X_l^t$ and $f^g = f_l^t$

3.4.9 Terminate

The *PSO* algorithm is terminated with a maximal number of iterations, the best particle vector of the entire swarm cannot be improved further after a sufficiently large number of iterations, or a maximum CPU time, otherwise go to step 2.

According the *gbest* model of Kennedy and Eberhart (2001) the pseudo code of the *PSO* algorithm is elaborated in the below figure.

```
Begin
     Initialize one particles positions
     Do{
     For each particle
           Calculate fitness value
           Find the particle best (Pi)
           Find the global best (G)
           For each particle
                Calculate particle velocity, V, according(3.3)
                Update particle position, X, according (3.2)
     }While (termination criterion is not met)
End
```

Figure 3.2 The pseudo code of PSO algorithm

3.5 THE EFFECTS OF THE PARAMETERS

3.5.1 Examples of dynamic behavior

Simulations of particle behavior for parameters *a* and *b* were performed with: $x_0 = 2$, $v_0 = -0.1$, p = 0, m = 50 iterations. (Trelea, 2003)

$$\vec{v}_{k+1} = \vec{a} \otimes \vec{v}_k + \vec{b}_1 \otimes \vec{r}_1 \otimes (\vec{p}_1 - \vec{x}_k) + \vec{b}_2 \otimes \vec{r}_2 \otimes (\vec{p}_2 - \vec{x}_k) \tag{3.6}$$

$$\vec{x}_{k+1} = \vec{c} \otimes \vec{x}_k + \vec{d} \otimes \vec{v}_{k+1} \tag{3.7}$$

The particle samples its state space relatively well. The exploration of state space and the exploitation of the current optimum are balanced. In contrast, the oscillations shown in Fig. 3.3(b) decay quickly. The exploitation is favored compared to exploration. As a general rule, parameter couples close to the center of the stability triangle induce quick convergence, while parameter couples close to its borders require

many iterations to converge. The terms "slow" and "quick" convergence should be related to the allowed number of iterations *(m)*. If, for example, m = 1000 iterations were allowed instead of 50, then the parameters used in Fig. 3.3(a) should be interpreted as inducing a "quick" convergence, since most of the particle positions *(*≈ 900*)* would be quite close to the equilibrium. In real-life problems the number of allowed iterations is a function of the admissible computation time and of the complexity of the cost function. Harmonic oscillations can be combined with zigzagging as in Fig. 3.3(c) (complex eigenvalues with negative real part).

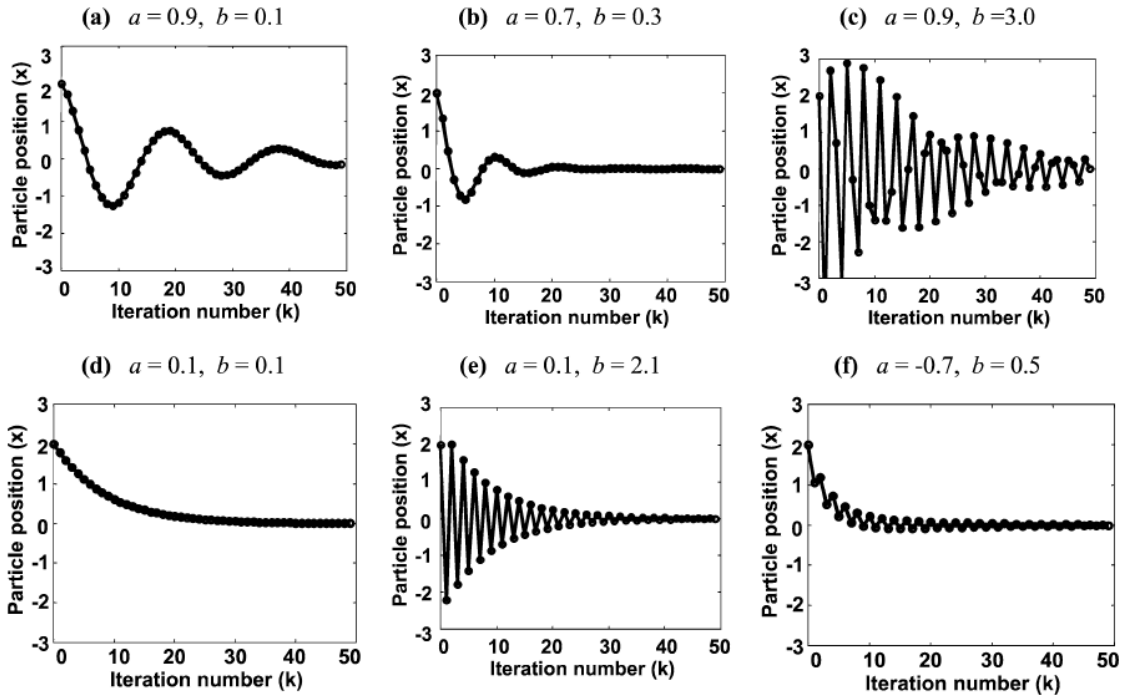

Figure 3.3. Examples of dynamic behavior of a single particle for various choices of the parameters a and b. (a) Harmonic oscillations with slow convergence. (b) Harmonic oscillations with quick convergence. (c) Harmonic oscillations with zigzagging. (d) Non-oscillatory convergence. (e) Symmetric zigzagging. (f) Asymmetric zigzagging.

An example of non-oscillatory convergence is given in Fig. 3.3(d) (real positive eigenvalues). For optimization purposes this behavior is not recommended in general, since the state space is only sampled on the one side of the current optimum. In special cases, however, this might be a useful option. For example, negative values of x might not make sense in a given optimization problem while the optimum is suspected to lie at or near zero.

Symmetric zigzagging convergence is shown in Fig. 3.3(e) (real negative eigenvalues). The parameters a and b can be tuned to make the convergence either slow

or fast as in the case of harmonic oscillations. Asymmetric zigzagging is illustrated in Fig. 3.3(f) (real eigenvalues with opposite signs).

3.5.2 Effect of the random numbers

The rigorous analysis of the optimization algorithm with random numbers described by Eqs. (3.6) and (3.7). Qualitatively, the considerations presented in the previous paragraphs remain valid, however, as shown by extensive simulation studies. The presence of random numbers enhances the zigzagging tendency and slows down convergence, thus improving the state space exploration and preventing premature convergence to non-optimal points. This is especially true when the particle's own attraction point p_1 is situated far from the population attraction point p_2. The equivalent attraction point p, is, in the case of the random algorithm, given by:

$$p = \frac{b_1 r_1}{b_1 r_1 + b_2 r_2} p_1 + \frac{b_2 r_2}{b_1 r_1 + b_2 r_2} p_2 \tag{3.8}$$

If $p_1 = p_2$, it changes from iteration to iteration even if no better solutions are discovered, i.e., p_1 and p_2 remain constant. In the long run, however, it is expected that $p_1 = p_2$ as all the particles in the population "agree" upon a single best point which becomes the unique attractor. In this case, Eq. (3.8) says that $p = p_1 = p_2$ irrespective of the generated random numbers.

CHAPTER 4

THE PROPOSED CONTINUOUS AND DISCRETE PSO ALGORITHMS FOR UFL PROBLEM

4.1 THE CONTINUOUS PSO ALGORITHM FOR UFL PROBLEM

The *Continuous PSO (CPSO)* algorithm proposed here for the *UFL* problem considers each particle based on three key vectors; position (X_i), velocity (V_i), and open facility (Y_i). $X_i = [x_{i1}, x_{i2}, x_{i3},\ldots,x_{in}]$ denotes the i^{th} position vector in the swarm, where x_{ik} is the position value of the i^{th} particle with respect to the k^{th} dimension $(k=1,2,3,\ldots,n)$. $V_i = [\ v_{i1}, v_{i2}, v_{i3},\ldots,v_{in}\ ]$ denotes the i^{th} velocity vector in the swarm, where v_{ik} is the velocity value of the i^{th} particle with respect to the k^{th} dimension. $Y_i = [y_{i1}, y_{i2}, y_{i3}\ldots y_{in}\ ]$ represents the opening or closing facilities identified based on the position vector(X_i), where y_{ik} represents opening or closing the k^{th} facility of the i^{th} particle. For an *n-facility* problem, each particle contains n number of dimensions.

Initially, the position and velocity vectors are generated randomly and uniformly as continuous sets of values, using the following rules:

$$v_{ij} = v_{\min} + (v_{\max} - v_{\min}) \times r_2 \tag{4.1}$$

$$x_{ij} = x_{\min} + (x_{\max} - x_{\min}) \times r_1 \tag{4.2}$$

where v_{min}=-4.0, v_{max}=4.0, x_{min}=-10.0, x_{max}=10.0 which are consistent with the literature (Shi and Eberhart, 1998). r_1 and r_2 are uniform random numbers between [0,1]. The position vector

$X_i = [x_{i1}, x_{i2}, x_{i3},\ldots,x_{in}]$ corresponds to the continuous position values for n facilities, but it does not represent a candidate solution to calculate a total cost. In order to create a candidate solution, a particle, the position vector is converted to binary

variables, $Y_i \leftarrow X_i$, which is also a key element of a particle. In other words, a continuous set is converted to a discrete set for the purpose of creating a candidate solution, particle. The fitness of the i^{th} particle is calculated by using open facility vector (Y_i). For simplicity, from now on $f_i(Y_i \leftarrow X_i)$ will be denoted with f_i.

In order to ascertain how to derive an open facility vector from a position vector, an instance of 5-facility problem is illustrated in Table 4.1. Position values are converted to binary variables using the following formula:

$$y_i = \lfloor\, |x_i \bmod 2|\, \rfloor \tag{4.3}$$

In equation (4.3) a position value is first divided by 2 and the absolute value of the remainder is floored; then the obtained integer number is taken as an element of the Y vector. For example, fifth element of Y vector, y_5, can be found as follows:

$$\lfloor\, |\text{-}5.45 \bmod 2|\, \rfloor = \lfloor\, |\text{-}1.45|\, \rfloor = \lfloor\, 1.45\, \rfloor = 1.$$

Table 4.1 An illustration of deriving open facility vector from position vector for a 5-facility problem for CPSO

i^{th} Particle Vectors	Particle Dimension (k)				
	1	2	3	4	5
Position Vector(X_i)	1.8	3.01	-0.99	0.72	-5.45
Velocity Vector(V_i)	-0.52	2.06	3.56	2.45	-1.44
Open Facility Vector (Y_i)	1	1	0	0	1

Considering the 5-facility to 6-customer example shown in Table 4.2, the total cost of supplying all customers from the open facilities that is determined by *Open Facility Vector (Y_i)* in Table 4.1 can be calculated as follows:

Total Cost = {Open facilities fixed costs (fc_i) + minimum cost of supply from open facilities, i, to customer j, (c_{ij}))}

= {(12+5+9) + (min(2,3,1) + min(0,5,12) + min(11,6,8) + min(19,18,13) + min(3,9,10) + min(4,7,0)} = {(26) + (1 + 0 + 6 + 13 + 3 + 0)} = {26 + 23} = {49}

Table 4.2 An example of 5-facility to 6-customer problem

Facility Locations		**1**	**2**	**3**	**4**	**5**
Fixed Cost		12	5	3	7	9
Customers	**1**	2	3	6	7	1
	2	0	5	8	4	12
	3	11	6	14	5	8
	4	19	18	21	16	13
	5	3	9	8	7	10
	6	4	7	9	6	0

For each particle in the swarm, a personal best, $P_i = [p_{i1}, p_{i2}, ..., p_{in}]$, is defined, whereby p_{ik} denotes the position value of the i^{th} personal best with respect to the k^{th} dimension. The personal bests are determined just after generating Y_i vectors and their corresponding fitness values. In every generation, t, the personal best of each particle is updated based on its position vector and fitness value. Regarding the objective function, $f_i(Y_i \leftarrow X_i)$, the fitness values for the personal best of the i^{th} particle, P_i, is denoted by $f_i^p = f(Y_i \leftarrow P_i)$ and obtained with (4.4).

$$f_i^p = \begin{cases} f_i^{p(t+1)} & \text{if } f_i^{p(t+1)} \le f_i^{p(t)} \\ f_i^{p(t)} & \text{otherwise} \end{cases} \qquad (4.4)$$

The personal best values are equal to position values (P_i=X_i) initially, where P_i = $[p_{i1}=x_{i1}, p_{i2}=x_{i2}, p_{i3}=x_{i3}, ..., p_{in}=x_{in}]$ and the fitness values of the personal bests are equal to the fitness of positions, $f_i^{pb} = f_i$.

Then, the best particle in the whole swarm is selected as the global best (the best particle in the whole swarm). $G= [g_1, g_2, g_3, ..., g_n]$ denotes the best position of the globally best particle achieved so far in the whole swarm. Therefore, the global best fitness, $f_g = f(Y \leftarrow G)$, can be obtained by using the equation (4.5)

At the beginning, global best fitness value is determined as the best of personal bests over the whole swarm, $f_g = \min\{f_i^{pb}\}$, with its corresponding position vector X_g,

which is to be used for $G=X_g$, where $G = [g_1=x_{g1}, g_2=x_{g2}, g_3=x_{g3}, \ldots, g_n=x_{gn}]$ and $Y_g = [y_{g1}, y_{g2}, y_{g3}, \ldots, y_{gn}]$ denotes the open facility vector of the global best found.

$$f_g = \begin{cases} f_g^{t+1} & \text{if } f_g^{t+1} \le f_g^t \\ f_g^t & \text{otherwise} \end{cases} \tag{4.5}$$

Afterwards, the velocity of each particle is updated based on its personal best and the global best in the following way:

$$v_{ik}^{t+1} = \left(w \cdot v_{ik}^t + c_1 r_1 \left(p_{ik}^t - x_{ik}^t\right) + c_2 r_2 \left(g_k^t - x_{ik}^t\right)\right) \tag{4.6}$$

where w is the inertia weight used to control the impact of the previous velocities on the current one and t stands for generation . In addition, r_1 and r_2 are random numbers in [0,1] and c_1 and c_2 are the learning factors, which are also called social and cognitive parameters. The next step is to update the positions in the following way.

$$x_{ik}^{t+1} = x_{ik}^t + v_{ik}^{t+1} \tag{4.7}$$

After obtaining position values updated for all particles, the corresponding open facility vectors are determined with their fitness values in order to start a new iteration if the predetermined stopping criterion is not satisfied.

```
Begin
     Initialize Velocity (4.1)
     Initialize Position (4.2)
     Obtain open facility vector (4.3)
     Evaluate (2.1)
     Do
          Find personal best (4.4)
          Find global best (4.5)
          Update Velocity (4.6)
          Update Position (4.7)
          Update open facility vector (4.3)
          Evaluate (2.1)
          Apply local search (for CPSO_LS)
     While (Not Termination)
End
```

Figure 4.1 Pseudo code of *CPSO* algorithm for *UFL* problem

In this study, we apply the *gbest* model of Kennedy and Eberhart (2001) for *CPSO*, which is elaborated in the above pseudo code.

4.2 THE DISCRETE PSO ALGORITHM FOR UFL PROBLEM

As usual, actual modeling begins with decision variables in optimization. However, these decisions qualitatively differ from each other in two ways: First, if it can take on any value in a specified interval that called continuous variable; the other, if it is limited to a fixed or countable set of values that called discrete variable. When there is an option, such as when optimal variable magnitudes are likely to be large enough that fractions have no practical importance, modeling with continuous variables is preferred to discrete because optimizations over continuous variables are generally more tractable than are ones over discrete variables (Rardin, 1998). In discrete variables, the choices are often only 1 and 0 namely all-or-nothing. Usually selecting, opening or affirmative decisions are represented by 1 and non-selecting, closing or negative decisions are represented by 0.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far (the fitness value is also stored). This value is denoted P_i. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by all particles in the population. This best value is a global best and called *G*.

The *DPSO* algorithm proposed here for the *UFL* problems considers each particle based on only open facility vector (Y_i). $Y_i = [y_{i1}, y_{i2}, y_{i3} \ldots y_{in}]$ represents the opening or closing facilities identified based on the position vector(X_i), where y_{ik} represents opening or closing k^{th} facility of the i^{th} particle. For an *n-facility* problem, each particle contains *n* number of dimensions. The dimensions of *Y* randomly generated as 1 or 0. The fitness of the i^{th} particle is calculated by using Y_i vector and denoted with $f_i(Y_i)$.

Since the behavior of a particle is a compromise among three possible choices: to follow its own position (Y_i^t), to go towards its personal best position P_i^t and to go

towards the best position of the particle in the whole swarm population G^t, the position of the particle at iteration *t* can be updated as follows (Pan et al., 2006).

$$Y_i^t = c_2 \oplus F_3\left(c_1 \oplus F_2\left(w \oplus F_1\left(Y_i^{t-1}\right), P_i^{t-1}\right), G^{t-1}\right) \tag{4.8}$$

$$\lambda_i^t = w \oplus F_1\left(Y_i^{t-1}\right) \tag{4.9}$$

The update equation (4.8) consists of three components: The first component is (4.9), which represents the *velocity* of the particle. In the (4.9), F_1 represents the swap operator which is conducted in such a way that two distinct facilities from the open facility vector, Y_i^{t-1}, of particle are randomly selected and interchanged with the probability of *w*. In other words, a uniform random number, *r,* is generated between 0 and 1. If *r* is less than *w* then the swap operator is applied to generate a perturbed Y_i vector of the particle by $\lambda_i^t = F_1\left(Y_i^{t-1}\right)$, otherwise current Y_i is kept as $\lambda_i^t = Y_i^{t-1}$.

$$\delta_i^t = c_1 \oplus F_2\left(\lambda_i^t, P_i^{t-1}\right) \tag{4.10}$$

The second component is (4.10), which is the *cognition* part of the particle representing the private thinking of the particle itself. In the component (4.10), F_2 represents the crossover operator conducted by using a one-cut crossover with the probability of c_1. Note that λ_i^t and P_i^{t-1} will be the first and second parents for the crossover operator respectively. It is resulted either in $\delta_i^t = F_2\left(\lambda_i^t, P_i^{t-1}\right)$ or in $\delta_i^t = \lambda_i^t$ depending on the choice of a uniform random number.

$$X_i^t = c_2 \oplus F_3\left(\delta_i^t, G^t\right) \tag{4.11}$$

The third component is (4.11), which is the *social* part of the particle representing the collaboration among particles. In the component (4.11), F_3 represents the crossover operator conducted by using a two-cut crossover with the probability of c_2. Note that δ_i^t and G^{t-1} will be the first and second parents for the crossover operator respectively. It is resulted either in $Y_i^t = F_3\left(\delta_i^t, G^{t-1}\right)$ or in $Y_i^t = \delta_i^t$ depending on the choice of a uniform random number (Pan et al., 2006).

The corresponding Y_i vectors are determined with their fitness values so as to start a new iteration if the predetermined stopping criterion is not satisfied.

In this study, we apply the *gbest* model of Kennedy and Eberhart (2001) for *DPSO,* which is elaborated in the following pseudo code given below.

```
Begin
     Initialize open facility vector
     Evaluate (2.1)
     Do
          Find personal best (4.4)
          Find global best (4.5)
          Update open facility vector (4.8)
               Apply velocity component (4.9)
               Apply cognition component (4.10)
               Apply social component (4.11)
          Evaluate (2.1)
          Apply local search (for DPSO_LS)
     While (Not Termination)
End
```

Figure 4.2 Pseudo code of *DPSO* algorithm for *UFL* problem

An example is presented to demonstrate how an open facility location vector gets the fitness of the solution. However, it does not show the steps of PSO algorithm.

Table 4.3 An illustration of open facility vector for a 5-facility problem for DPSO

i^{th} Particle Vectors	Particle Dimension (k)				
	1	2	3	4	5
Open Facility Vector (Y_i)	1	1	0	0	1

Considering the 5-facility problem shown in Table 4.2, total cost of the Open facility vector (Y_i) can be calculated as follows:

Fixed Costs+ min (cost of which facility supply the customer) =

{(12+5+9) + (min(2,3,1) + min(0,5,12) + min(11,6,8) + min(19,18,13) + min(3,9,10) + min(4,7,0)} = {(26) + (1 + 0 + 6 + 13 + 3 + 0)} = {26 + 23} = {49}

4.3 THE LOCAL SEARCH ALGORITHM FOR PSO

Apparently, *PSO* conducts such a rough search that it produces premature results, which do not offer satisfactory solutions. For this purpose, it is inevitable to hybridize *PSO* with a local search algorithm so as to produce more satisfactory solutions. In this study, a simple local search method for both *CPSO* and *DPSO* is applied.

The solutions obtained from the metaheuristics methods can be improved by hybridizing the algorithm with local search. The neighborhood structure with which neighbor solutions are determined to move is one of the key elements in metaheuristics. The algorithm starts with a complete solution and tries to find a better solution by using the neighborhood of the current solution. The solutions are called as neighbors if the latter solution can be obtained by modifying the current one.

The performance of the hybrid algorithm depends on the efficiency of the neighborhood structure. Thus, a local search method to neighbors of the global best position vector is proposed. For the *UFL* problem, flip operator is employed as a neighborhood structure. Flip operator can be defined as picking one position value of the global best randomly and then changing its value with using (4.12) for $CPSO_{LS}$.

$$g_i = \begin{cases} 0 \le \rho \le 0.5 & g_i + 1 \\ 0.5 < \rho \le 1 & g_i - 1 \end{cases} \tag{4.12}$$

where ρ is a uniformly generated number between 0 and 1. To employ same operator for $DPSO_{LS}$ (4.13) is used since only binary values are stored in vectors. This operator is used for opening a new facility or closing an open one.

$$g_i \leftarrow 1 - g_i \tag{4.13}$$

The local search algorithm applied in this study is sketched in Figure 4.3. The global best found at the end of each iteration of *PSO* is adopted as the initial solution by local search algorithm. In order not to loss the best found and to diversify the solution, the global best is randomly modified in which two facilities are flipped based on both random parameters generated, η and κ. Then, flip operator is applied to global best vector as long as it gets a better solution. After evaluating, the final produced solution, *s***,** is replaced with the old global best if it is better than the initial one.

```
Begin
    Set global best position vector (Y_g) to s_0
    Modify s_0 based on η,κ and set to s
    Set 0 to loop
    Repeat:
         Apply Flip to s and get s_1
         if(f(s_1)• f(s))
         Replace s with s_1
         else
         loop=loop+1
    Until loop< n is false
    if (f(s)• f(s_0))
         Replace Y_g with s
End
```

Figure 4.3 Pseudo code for local search

CHAPTER 5

EXPERIMENTAL RESULTS

This experimental study has been completed in two stages; first, we compared the *PSO* algorithms without local search then *hybrid PSO* algorithms statistically and with respect to their solution quality. Experimental results provided in this section are carried out with four algorithms over 15 benchmark problems well-known by the researchers of *UFL* field. The benchmarks are undertaken from the *OR Library* (Beasley, 2001), a collection of benchmarks for operations research *(OR)* studies. The benchmarks are introduced in Table 5.1 with their sizes and the optimum values. Although the optimum values are known, it is really hard to hit the optima in every attempt of optimization. Since the main idea is to test the performance of PSO algorithm with *UFL* benchmark, the results are provided in Tables as the solution quality: *Average Relative Percent Error (ARPE), Hit to optimum Rate (HR)* and *Computational Processing Time (CPU). ARPE* is the percentage of difference from the optimum and defined as following:

$$ARPE = \sum_{i=1}^{R} \left(\frac{H_i - U}{U} \right) \times 100 \Big/ R \qquad \textbf{(5.1)}$$

where H_i denotes the i^{th} replication solution value, U is the optimal value provided in the literature and R is the number of replications. *HR* provides the ratio between the number of runs yielded the optimum and the total numbers of experimental trials.

Obviously, the higher the HR the better quality of solution, while the lower the ARPE the better quality. The computational time spent for CPSO and DPSO cases are obtained as time to get best value over 1000 iterations, while for CPSOLS and DPSOLS cases are obtained as time to get best value over 250 iterations. All algorithms and other

related software were coded with Borland C++ Builder 6 and run on an Intel Centrino 1.7 GHz PC with 512MB memory.

Table 5.1 Benchmarks tackled with the sizes (number of facilities × number of customers) and the optimum fitness values

Problem	Size	Optimum
Cap71	16×50	932615.75
Cap72	16×50	977799.40
Cap73	16×50	1010641.45
Cap74	16×50	1034976.98
Cap81	25×50	796648.44
Cap82	25×50	854704.20
Cap83	25×50	893782.11
Cap84	25×50	928941.75
Cap91	25×50	796648.44
Cap92	25×50	854704.20
Cap93	25×50	893782.11
Cap94	25×50	928941.75
Cap101	25×50	796648.44
Cap102	25×50	854704.20
Cap103	25×50	893782.11
Cap104	25×50	928941.75
Cap111	50×50	793439.56
Cap112	50×50	851495.33
Cap113	50×50	893076.71
Cap114	50×50	928941.75
Cap121	50×50	793439.56
Cap122	50×50	851495.33
Cap123	50×50	893076.71
Cap124	50×50	928941.75
Cap131	50×50	793439.56
Cap132	50×50	851495.33
Cap133	50×50	893076.71
Cap134	50×50	928941.75
CapA	100×1000	17156454.48
CapB	100×1000	12979071.58
CapC	100×1000	11505594.33

5.1 CONTINUOUS PSO PARAMETERS

The parameters used for the *CPSO* and $CPSO_{LS}$ algorithms are as follows: The size of the population (swarm) is the number of facilities, the social and cognitive parameters are taken as $c_1=c_2=2$ and other parameters: minimum position value, x_{min}= -

10.0, maximum position value, x_{max}=10.0, minimum velocity v_{min}=-4.0, maximum velocity v_{max}=4.0 are set as shown which are consistent with the literature (Shi and Eberhart, 1998). Inertia weight, *w,* is taken as a random number between 0.5 and 1. As mentioned before, number of iteration is set to 1000 iterations for each benchmark suite for *CPSO* algorithm and 250 iterations for $CPSO_{LS}$ algorithm. Finally, each problem solution run is conducted for 30 replications. All replications fitness results and statistical CPU results are given in Appendix.

5.2 DISCRETE PSO PARAMETERS

There are fewer parameters used for the *DPSO* and $DPSO_{LS}$ algorithms and they are as follows: The size of the population (swarm) is the number of facilities, the social and cognitive probabilities, c_1 and c_2, are set as c_1=c_2=0.5 and inertia weight, *w,* is set to 0,9. As it is mentioned before, number of iteration is set to 1000 iterations for each benchmark suite for *DPSO* algorithm and 250 iterations for $DPSO_{LS}$ algorithm. Each problem solution run is conducted for 30 replications. All replications fitness results and statistical CPU results are given in Appendix.

There are two termination criteria that have been applied for every run: First one is getting the optimum solution, the other is reaching the maximum iteration number that is chosen for obtain the result in a reasonable *CPU* time.

5.3 THE COMPARISON OF CPSO AND DPSO

The performance of *CPSO* algorithm looks not very impressive as the results produced within the range of time over 1000 iterations. The *CPSO* search found 6 optimal solutions whereas the *DPSO* algorithm found 12 among 15 benchmark problems. The *ARPE* index which is expected lower for good solution quality is very high for *CPSO* when applied *CapA, CapB* and *CapC* benchmarks and none of the attempts for these benchmarks hit the optimum value. As come to the *ARPE* index of *DPSO*, it is better than the *ARPE* index of *CPSO* but not satisfactory as expected. In term of *CPU*, *DPSO* is better than *CPSO* as well. It may be possible to improve the solutions quality by carrying on with algorithms for a further number of iterations, but,

then the main idea and useful motivation of employing the heuristics, i.e. getting a better quality within shorter time, will be lost. This fact imposed that it is essential to empower *PSO* with hybridizing with a local search algorithm. Thus a simple local search algorithm is employed in this case for that purpose, as mentioned before.

Table 5.2 Experimental Results of CPU and Fitness values for CPSO

CPSO	CPU	Fitness				
Problem	Average	Best	Worse	Std	ARPE	HR
Cap71	0.1318	932615.75	934199.14	562.23	0.03	0.83
Cap72	0.1318	977799.40	983713.81	1324.30	0.05	0.83
Cap73	0.1865	1010641.45	1012643.69	702.13	0.03	0.73
Cap74	0.1781	1034976.98	1045342.23	2124.54	0.10	0.00
Cap81	0.8823	796648.44	802457.23	1480.72	0.18	0.00
Cap82	0.7672	854704.20	857380.85	1015.64	0.13	0.33
Cap83	1.2474	893782.11	899348.08	1303.87	0.09	0.00
Cap84	0.6000	928941.75	944394.83	3842.64	0.29	0.60
Cap91	0.8813	796648.44	802457.23	1480.72	0.18	0.00
Cap92	0.7667	854704.20	857380.85	1015.64	0.13	0.33
Cap93	1.2516	893782.11	899348.08	1303.87	0.09	0.00
Cap94	0.6021	928941.75	944394.83	3842.64	0.29	0.60
Cap101	0.8818	796648.44	802457.23	1480.72	0.18	0.00
Cap102	0.7667	854704.20	857380.85	1015.64	0.13	0.33
Cap103	0.9938	893782.11	899424.91	1695.79	0.14	0.00
Cap104	0.6026	928941.75	944394.83	3842.64	0.29	0.60
Cap111	3.6182	795291.86	804549.64	2429.54	0.91	0.00
Cap112	3.5474	851495.33	868567.16	4297.07	0.76	0.00
Cap113	3.9031	893076.71	909908.70	4753.48	0.55	0.00
Cap114	3.3375	928941.75	951803.25	6619.05	0.69	0.23
Cap121	3.6141	795291.86	804549.64	2429.54	0.91	0.00
Cap122	3.5495	851495.33	868567.16	4297.07	0.76	0.00
Cap123	3.8901	893076.71	909908.70	4753.48	0.55	0.00
Cap124	3.3359	928941.75	951803.25	6619.05	0.69	0.23
Cap131	3.6156	795291.86	804549.64	2429.54	0.91	0.00
Cap132	3.5599	851495.33	868567.16	4297.07	0.76	0.00
Cap133	3.7792	893076.71	909908.70	4210.93	0.50	0.00
Cap134	3.3333	928941.75	951803.25	6619.05	0.69	0.23
CapA	19.5739	18351465.40	24638983.70	1608650.32	21.24	0.00
CapB	17.1318	13390061.37	15356618.36	532161.62	10.14	0.00
CapC	17.6149	11970113.17	13572876.10	350412.47	8.16	0.00

Table 5.3 Experimental Results of CPU and Fitness values for DPSO

DPSO	CPU	Fitness				
Problem	Average	Best	Worse	Std	ARPE	HR
Cap71	0.0641	932615.75	932615.75	0.00	0.00	1.00
Cap72	0.0651	977799.40	977799.40	0.00	0.00	1.00
Cap73	0.0708	1010641.45	1010641.45	0.00	0.00	1.00
Cap74	0.0693	1034976.98	1034976.98	0.00	0.00	1.00
Cap81	0.3130	796648.44	796648.44	0.00	0.00	1.00
Cap82	0.3062	854704.20	854704.20	0.00	0.00	1.00
Cap83	0.9365	893782.11	893782.11	0.00	0.00	1.00
Cap84	0.2011	928941.75	928941.75	0.00	0.00	1.00
Cap91	0.3119	796648.44	796648.44	0.00	0.00	1.00
Cap92	0.3063	854704.20	854704.20	0.00	0.00	1.00
Cap93	0.9333	893782.11	893782.11	0.00	0.00	1.00
Cap94	0.2016	928941.75	928941.75	0.00	0.00	1.00
Cap101	0.3130	796648.44	796648.44	0.00	0.00	1.00
Cap102	0.3062	854704.20	854704.20	0.00	0.00	1.00
Cap103	0.3625	893782.11	894008.14	57.35	0.00	0.93
Cap104	0.2021	928941.75	928941.75	0.00	0.00	1.00
Cap111	2.6141	793439.56	795927.69	759.45	0.16	0.13
Cap112	2.6589	851495.33	854061.12	683.00	0.09	0.17
Cap113	2.7984	893076.71	894095.76	420.88	0.04	0.47
Cap114	1.7136	928941.75	928941.75	0.00	0.00	1.00
Cap121	2.5453	793439.56	795935.81	822.12	0.17	0.13
Cap122	2.6318	851495.33	854061.12	683.00	0.09	0.17
Cap123	2.8167	893076.71	894095.76	420.88	0.04	0.47
Cap124	1.7146	928941.75	928941.75	0.00	0.00	1.00
Cap131	2.5464	793439.56	795935.81	822.12	0.17	0.13
Cap132	2.6328	851495.33	854061.12	683.00	0.09	0.17
Cap133	2.5292	893076.71	894095.76	398.07	0.04	0.43
Cap134	1.7167	928941.75	928941.75	0.00	0.00	1.00
CapA	17.8972	17810901.35	19366467.31	367959.31	8.65	0.00
CapB	17.0652	13287703.69	13923731.30	135262.96	4.92	0.00
CapC	17.1340	11855107.73	12180785.60	85472.59	4.54	0.00

5.3.1 Statistically Testing

To compare the results statistically and to understand which algorithm outperforms the other algorithm the paired *t-test* method is employed. Because different pairs are independent, the differences (D) of each replication are independent of one another. If we let D= X-Y, where X and Y are the first and second observations, respectively, within an arbitrary pair, then the expected difference is: $\mu_D = E(X-Y) = E(X) - E(Y) = \mu_1 - \mu_2$. Thus any hypothesis about $\mu_1 - \mu_2$ can be phrased as a hypothesis about the mean difference μ_D. That is, to test hypotheses about

$\mu_1 - \mu_2$ when data is paired, the differences D_1, D_2, …, D_n are formed and a one-sample *t-test* (based on n-1 *df*) on the differences is carried out (Devore, 2000).

$$H_0 : \mu_D = \Delta_0 \tag{5.2}$$

$$t = \frac{\overline{d} - \Delta_0}{s_D / \sqrt{n}} \tag{5.3}$$

where H_0 is a null hypothesis and $\mu_D = \mu_1 - \mu_2$, t is test statistic value.

$\overline{d}$ and s_D are the sample mean and standard deviation, respectively, of the d_i's.

Table 5.4 Statistic comparison between CPSO and DPSO

CPSO-DPSO		Level of significance:		$t_{0.05, 29}$	$t_{0.010, 29}$	$t_{0.005, 29}$	$t_{0.0005, 29}$
Problem	$\overline{d}$	$\overline{s_D}$	t-value	1.6991	2.462	2.7564	3.6594
Cap71	242.890	562.234	2.366	Accept	Deny	Deny	Deny
Cap72	487.713	1324.303	2.017	Accept	Deny	Deny	Deny
Cap73	345.448	702.130	2.695	Accept	Accept	Deny	Deny
Cap74	984.860	2124.542	2.539	Accept	Accept	Deny	Deny
Cap81	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap82	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap83	847.699	1303.867	3.561	Accept	Accept	Accept	Deny
Cap84	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap91	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap92	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap93	847.699	1303.867	3.561	Accept	Accept	Accept	Deny
Cap94	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap101	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap102	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap103	1278.946	1703.947	4.111	Accept	Accept	Accept	Accept
Cap104	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap111	5945.298	2598.722	12.531	Accept	Accept	Accept	Accept
Cap112	5676.582	4496.727	6.914	Accept	Accept	Accept	Accept
Cap113	4473.693	4609.130	5.316	Accept	Accept	Accept	Accept
Cap114	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
Cap121	5850.919	2604.797	12.303	Accept	Accept	Accept	Accept
Cap122	5676.582	4496.727	6.914	Accept	Accept	Accept	Accept
Cap123	4473.693	4609.130	5.316	Accept	Accept	Accept	Accept
Cap124	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
Cap131	5850.919	2604.797	12.303	Accept	Accept	Accept	Accept
Cap132	5676.582	4496.727	6.914	Accept	Accept	Accept	Accept
Cap133	4055.578	4136.184	5.370	Accept	Accept	Accept	Accept
Cap134	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
CapA	2159683.544	1679082.732	7.045	Accept	Accept	Accept	Accept
CapB	677197.294	537134.664	6.905	Accept	Accept	Accept	Accept
CapC	416177.758	354427.972	6.431	Accept	Accept	Accept	Accept
	Number of accepted hypotheses *(H_1)*			31	29	27	25
	Number of denied hypotheses *(H_1)*			0	2	4	6

The results are investigated with 95%, 99%, 99.5%, 99.95% levels of confidence. The first proposed hypothesis *(H_1)* is DPSO is better than CPSO. If the H_1 is true, t-values for the problems have to be greater than the level of significance value. The *DPSO* produced significantly better fitness results than *CPSO* for all problems for 95% level of confidence. Apparently, for the other levels of confidence, number of accepted hypotheses is much more than denied hypotheses (Table 5.4). Thus, we can say that DPSO is generated more robust results than CPSO.

5.4 THE COMPARISON OF $CPSO_{LS}$ AND $DPSO_{LS}$

The performance of *PSO* algorithms with local searches looks very impressive compared to the *CPSO* and the *DPSO* algorithms with respect of all three indexes. HR is 1.00 which means 100% of the runs yield with optimum for all benchmark except *CapB* and *CapC* for $CPSO_{LS}$ and except *CapA, CapB* and *CapC* for $DPSO_{LS}$. On the other hand, it should be mentioned that $DPSO_{LS}$ found optimum solutions for all instances while $DPSO_{LS}$ found optimums except for *CapC*. The *ARPE* index results of $CPSO_{LS}$ and $DPSO_{LS}$ are very small for both algorithms and very similar to each other thus there is no meaningful difference. In term of *CPU,* $CPSO_{LS}$ consumed 18% more time than $DPSO_{LS}$ thus we can say that the results of $DPSO_{LS}$ are more robust than $CPSO_{LS}$.

Table 5.5 Experimental Results of CPU and Fitness values for $CPSO_{LS}$

CPSO_LS	CPU	Fitness				
Problem	Average	Best	Worse	Std	ARPE	HR
Cap71	0.0146	932615.75	932615.75	0.00	0.00	1.00
Cap72	0.0172	977799.40	977799.40	0.00	0.00	1.00
Cap73	0.0281	1010641.45	1010641.45	0.00	0.00	1.00
Cap74	0.0182	1034976.98	1034976.98	0.00	0.00	1.00
Cap81	0.1875	796648.44	796648.44	0.00	0.00	1.00
Cap82	0.0896	854704.20	854704.20	0.00	0.00	1.00
Cap83	0.2151	893782.11	893782.11	0.00	0.00	1.00
Cap84	0.0370	928941.75	928941.75	0.00	0.00	1.00
Cap91	0.1875	796648.44	796648.44	0.00	0.00	1.00
Cap92	0.0896	854704.20	854704.20	0.00	0.00	1.00
Cap93	0.2151	893782.11	893782.11	0.00	0.00	1.00
Cap94	0.0370	928941.75	928941.75	0.00	0.00	1.00
Cap101	0.1880	796648.44	796648.44	0.00	0.00	1.00
Cap102	0.0906	854704.20	854704.20	0.00	0.00	1.00
Cap103	0.2151	893782.11	893782.11	0.00	0.00	1.00
Cap104	0.0370	928941.75	928941.75	0.00	0.00	1.00
Cap111	1.4271	793439.56	793439.56	0.00	0.00	1.00
Cap112	1.0245	851495.33	851495.33	0.00	0.00	1.00
Cap113	1.3651	893076.71	893076.71	0.00	0.00	1.00
Cap114	0.3635	928941.75	928941.75	0.00	0.00	1.00
Cap121	1.4276	793439.56	793439.56	0.00	0.00	1.00
Cap122	1.0240	851495.33	851495.33	0.00	0.00	1.00
Cap123	1.3651	893076.71	893076.71	0.00	0.00	1.00
Cap124	0.3635	928941.75	928941.75	0.00	0.00	1.00
Cap131	1.4281	793439.56	793439.56	0.00	0.00	1.00
Cap132	1.0245	851495.33	851495.33	0.00	0.00	1.00
Cap133	1.3651	893076.71	893076.71	0.00	0.00	1.00
Cap134	0.3635	928941.75	928941.75	0.00	0.00	1.00
CapA	22.3920	17156454.48	17346752.16	34743.44	0.04	0.97
CapB	30.6541	12979071.58	13084984.12	39458.67	0.33	0.40
CapC	27.4234	11509361.66	11580061.89	16072.22	0.09	0.00

Table 5.6 Experimental Results of CPU and Fitness values for $DPSO_{LS}$

DPSO_LS	CPU	Fitness				
Problem	Average	Best	Worse	Std	ARPE	HR
Cap71	0.0130	932615.75	932615.75	0.00	0.00	1.00
Cap72	0.0078	977799.40	977799.40	0.00	0.00	1.00
Cap73	0.0203	1010641.45	1010641.45	0.00	0.00	1.00
Cap74	0.0109	1034976.98	1034976.98	0.00	0.00	1.00
Cap81	0.1516	796648.44	796648.44	0.00	0.00	1.00
Cap82	0.0557	854704.20	854704.20	0.00	0.00	1.00
Cap83	0.1693	893782.11	893782.11	0.00	0.00	1.00
Cap84	0.0339	928941.75	928941.75	0.00	0.00	1.00
Cap91	0.1500	796648.44	796648.44	0.00	0.00	1.00
Cap92	0.0562	854704.20	854704.20	0.00	0.00	1.00
Cap93	0.1693	893782.11	893782.11	0.00	0.00	1.00
Cap94	0.0344	928941.75	928941.75	0.00	0.00	1.00
Cap101	0.1505	796648.44	796648.44	0.00	0.00	1.00
Cap102	0.0557	854704.20	854704.20	0.00	0.00	1.00
Cap103	0.1693	893782.11	893782.11	0.00	0.00	1.00
Cap104	0.0344	928941.75	928941.75	0.00	0.00	1.00
Cap111	0.9927	793439.56	794299.85	157.07	0.00	0.97
Cap112	0.7750	851495.33	851495.33	0.00	0.00	1.00
Cap113	1.0510	893076.71	893251.51	31.91	0.00	0.97
Cap114	0.5594	928941.75	928941.75	0.00	0.00	1.00
Cap121	0.9932	793439.56	794299.85	157.07	0.00	0.97
Cap122	0.7740	851495.33	851495.33	0.00	0.00	1.00
Cap123	1.0510	893076.71	893251.51	31.91	0.00	0.97
Cap124	0.5594	928941.75	928941.75	0.00	0.00	1.00
Cap131	0.9922	793439.56	794299.85	157.07	0.00	0.97
Cap132	0.7745	851495.33	851495.33	0.00	0.00	1.00
Cap133	1.0516	893076.71	893251.51	31.91	0.00	0.97
Cap134	0.5594	928941.75	928941.75	0.00	0.00	1.00
CapA	19.5889	17156454.48	17844165.28	243932.83	1.51	0.33
CapB	19.6359	12979071.58	13444180.76	100126.81	0.86	0.20
CapC	18.7685	11505594.33	11716349.47	49717.56	0.37	0.13

5.4.1 Statistically Testing

To compare the results between proposed algorithms statistically with using the *t-test* with 95%, 99%, 99.5%, 99.95% levels of confidence more 3 hypotheses is suggested. The second proposed hypothesis *(H_2)* is $CPSO_{LS}$ is better than CPSO. If the H_2 is true, t-values for the problems have to be greater than the level of significance value. The $CPSO_{LS}$ produced significantly better fitness results than *CPSO* for all problems for 95% level of confidence. Apparently, for the other levels of confidence, number of accepted hypotheses is much more than denied hypotheses (Table 5.7). Thus, we can say that $CPSO_{LS}$ is generated more robust results than CPSO.

Table 5.7 Statistic comparison between CPSO and $CPSO_{LS}$

$CPSO_{LS}$-CPSO		Level of significance		$t_{0.05, 29}$	$t_{0.010, 29}$	$t_{0.005, 29}$	$t_{0.0005, 29}$
Problem	Av. Dev	Std. Dev.	t-value	1.699	2.462	2.756	3.659
Cap71	242.890	562.234	2.366	Accept	Deny	Deny	Deny
Cap72	487.713	1324.303	2.017	Accept	Deny	Deny	Deny
Cap73	345.448	702.130	2.695	Accept	Accept	Deny	Deny
Cap74	984.860	2124.542	2.539	Accept	Accept	Deny	Deny
Cap81	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap82	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap83	847.699	1303.867	3.561	Accept	Accept	Accept	Deny
Cap84	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap91	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap92	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap93	847.699	1303.867	3.561	Accept	Accept	Accept	Deny
Cap94	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap101	1458.365	1480.723	5.395	Accept	Accept	Accept	Accept
Cap102	1150.032	1015.636	6.202	Accept	Accept	Accept	Accept
Cap103	1294.015	1695.785	4.180	Accept	Accept	Accept	Accept
Cap104	2660.544	3842.637	3.792	Accept	Accept	Accept	Accept
Cap111	7227.427	2429.538	16.294	Accept	Accept	Accept	Accept
Cap112	6440.656	4297.071	8.210	Accept	Accept	Accept	Accept
Cap113	4873.163	4753.479	5.615	Accept	Accept	Accept	Accept
Cap114	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
Cap121	7227.427	2429.538	16.294	Accept	Accept	Accept	Accept
Cap122	6440.656	4297.071	8.210	Accept	Accept	Accept	Accept
Cap123	4873.163	4753.479	5.615	Accept	Accept	Accept	Accept
Cap124	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
Cap131	7227.427	2429.538	16.294	Accept	Accept	Accept	Accept
Cap132	6440.656	4297.071	8.210	Accept	Accept	Accept	Accept
Cap133	4429.388	4210.926	5.761	Accept	Accept	Accept	Accept
Cap134	6422.655	6619.049	5.315	Accept	Accept	Accept	Accept
CapA	3638032.944	1602699.056	12.433	Accept	Accept	Accept	Accept
CapB	1273040.211	552455.019	12.621	Accept	Accept	Accept	Accept
CapC	928543.295	353184.171	14.400	Accept	Accept	Accept	Accept
	Number of accepted hypotheses *(H_2)*			31	29	27	25
	Number of denied hypotheses *(H_2)*			0	2	4	6

The third proposed hypothesis *(H_3)* is $DPSO_{LS}$ is better than DPSO. If the H3 is true, t-values for the problems have to be greater than the level of significance value. Since, both DPSO and $DPSO_{LS}$ hit the optimum for all replications for the problems Cap71-74, Cap81-84, Cap91-94, Cap101-102, Cap104, Cap 114, Cap124 and Cap134 there is no deviation between the results of DPSO and $DPSO_{LS}$. Thus, t-value can not be calculated for these problems. For the other problems except Cap103 the $DPSO_{LS}$ produced significantly better fitness results than *DPSO* for all 95%, 99%, 99.5%,

99.95% levels of confidence (Table 5.8). To conclude, we can say that $DPSO_{LS}$ is generated more robust results than DPSO.

Table 5.8 Statistic comparison between DPSO and $DPSO_{LS}$ with *t*-test

DPSO-$DPSO_{LS}$		Level of significance		$t_{0.05,\ 29}$	$t_{0.010,\ 29}$	$t_{0.005,\ 29}$	$t_{0.0005,\ 29}$
Problem	Av. Dev	Std. Dev.	t-value	1.699	2.462	2.756	3.659
Cap71	0.000	0.000	-	-	-	-	-
Cap72	0.000	0.000	-	-	-	-	-
Cap73	0.000	0.000	-	-	-	-	-
Cap74	0.000	0.000	-	-	-	-	-
Cap81	0.000	0.000	-	-	-	-	-
Cap82	0.000	0.000	-	-	-	-	-
Cap83	0.000	0.000	-	-	-	-	-
Cap84	0.000	0.000	-	-	-	-	-
Cap91	0.000	0.000	-	-	-	-	-
Cap92	0.000	0.000	-	-	-	-	-
Cap93	0.000	0.000	-	-	-	-	-
Cap94	0.000	0.000	-	-	-	-	-
Cap101	0.000	0.000	-	-	-	-	-
Cap102	0.000	0.000	-	-	-	-	-
Cap103	15.069	57.346	1.439	Deny	Deny	Deny	Deny
Cap104	0.000	0.000	-	-	-	-	-
Cap111	1253.453	791.490	8.674	Accept	Accept	Accept	Accept
Cap112	764.074	682.998	6.127	Accept	Accept	Accept	Accept
Cap113	393.643	427.757	5.040	Accept	Accept	Accept	Accept
Cap114	0.000	0.000	-	-	-	-	-
Cap121	1347.831	855.087	8.633	Accept	Accept	Accept	Accept
Cap122	764.074	682.998	6.127	Accept	Accept	Accept	Accept
Cap123	393.643	427.757	5.040	Accept	Accept	Accept	Accept
Cap124	0.000	0.000	-	-	-	-	-
Cap131	1347.831	855.087	8.633	Accept	Accept	Accept	Accept
Cap132	764.074	682.998	6.127	Accept	Accept	Accept	Accept
Cap133	367.983	404.950	4.977	Accept	Accept	Accept	Accept
Cap134	0.000	0.000	-	-	-	-	-
CapA	1224952.223	495059.638	13.553	Accept	Accept	Accept	Accept
CapB	526752.304	154216.787	18.708	Accept	Accept	Accept	Accept
CapC	480395.117	106686.599	24.663	Accept	Accept	Accept	Accept
	Number of accepted hypotheses *(H_3)*			12	12	12	12
	Number of denied hypotheses *(H_3)*			1	1	1	1

The fourth proposed hypothesis *(H_4)* is $DPSO_{LS}$ is better than $CPSO_{LS}$. If the H_4 is true, *t*-values for the problems have to be greater than the level of significance value. Since, both DPSO and $DPSO_{LS}$ hit the optimum for all replications for the problems Cap71-74, Cap81-84, Cap91-94, Cap101-104, Cap 112, Cap114, Cap 122, Cap124, Cap 132 and Cap134 there is no deviation between the results of $DPSO_{LS}$ and $CPSO_{LS}$. Thus, t-value can not be calculated for these problems. For the problems CapA, CapB

and CapC $DPSO_{LS}$ produced significantly better fitness results than $CPSO_{LS}$ for 95%, 99%, 99.5%levels of confidence (Table 5.9). For the 99.95% level of confidence H4 is not true for all problems except CapA. For the problems Cap111, Cap113, Cap121, Cap123, Cap131 and Cap133 $DPSO_{LS}$ produced worse fitness results than $CPSO_{LS}$ for all levels of confidence thus H_4 is denied for these problems (Table 5.9). To conclude, we can not say that $DPSO_{LS}$ is generated more robust results than $CPSO_{LS}$.

Table 5.9 Statistic comparison between $DPSO_{LS}$ and $CPSO_{LS}$ with *t*-test

$DPSO_{LS}$- $CPSO_{LS}$		Level of significance		$t_{0.05, 29}$	$t_{0.010, 29}$	$t_{0.005, 29}$	$t_{0.0005, 29}$
Problem	Av. Dev	Std. Dev.	t-value	1.699	2.462	2.756	3.659
Cap71	0.000	0.000	-	-	-	-	-
Cap72	0.000	0.000	-	-	-	-	-
Cap73	0.000	0.000	-	-	-	-	-
Cap74	0.000	0.000	-	-	-	-	-
Cap81	0.000	0.000	-	-	-	-	-
Cap82	0.000	0.000	-	-	-	-	-
Cap83	0.000	0.000	-	-	-	-	-
Cap84	0.000	0.000	-	-	-	-	-
Cap91	0.000	0.000	-	-	-	-	-
Cap92	0.000	0.000	-	-	-	-	-
Cap93	0.000	0.000	-	-	-	-	-
Cap94	0.000	0.000	-	-	-	-	-
Cap101	0.000	0.000	-	-	-	-	-
Cap102	0.000	0.000	-	-	-	-	-
Cap103	0.000	0.000	-	-	-	-	-
Cap104	0.000	0.000	-	-	-	-	-
Cap111	28.676	157.067	1.000	Deny	Deny	Deny	Deny
Cap112	0.000	0.000	-	-	-	-	-
Cap113	5.827	31.914	1.000	Deny	Deny	Deny	Deny
Cap114	0.000	0.000	-	-	-	-	-
Cap121	28.676	157.067	1.000	Deny	Deny	Deny	Deny
Cap122	0.000	0.000	-	-	-	-	-
Cap123	5.827	31.914	1.000	Deny	Deny	Deny	Deny
Cap124	0.000	0.000	-	-	-	-	-
Cap131	28.676	157.067	1.000	Deny	Deny	Deny	Deny
Cap132	0.000	0.000	-	-	-	-	-
Cap133	5.827	31.914	1.000	Deny	Deny	Deny	Deny
Cap134	0.000	0.000	-	-	-	-	-
CapA	253397.177	236736.432	5.863	Accept	Accept	Accept	Accept
CapB	69090.613	107199.867	3.530	Accept	Accept	Accept	Deny
CapC	31970.420	49960.869	3.505	Accept	Accept	Accept	Deny
	Number of accepted hypotheses *(H_4)*			3	3	3	1
	Number of denied hypotheses *(H_4)*			6	6	6	8

CHAPTER 6

CONCLUSIONS

Developing solution methods for the *UFL* problem has been a hot topic of research for the last 40 years. Thus, *UFL* problems have been studied and examined extensively by various attempts and approaches. Because the *UFL* problem is NP-hard exact algorithms may not be able to solve large practical problems. Since they are NP-hard problems, the larger the size of the problem, the harder to find the optimal solution and furthermore, the longer to reach reasonable results. All important approaches relevant to *UFL* problems can be classified into two main categories: exact algorithms and metaheuristic based methods.

PSO is one of the latest metaheuristic methods in the literature. Based on the metaphor of social interaction and communication such as bird flocking and fish schooling, *PSO* was first introduced to optimize various continuous nonlinear functions by Eberhart and Kennedy (1995).

It is obvious that since the main decision in *UFL* is opening or closing facilities, *UFL* problems are classified in discrete problems. On the other hand, *PSO* is mainly designed for continuous problem thus it has some drawbacks when applying *PSO* for a discrete problem. Since *PSO* is developed for continuous optimization problem initially, most existing *PSO* applications are resorting to continuous function value optimization (Eberhart and Shi, 1998; Kennedy and Eberhart, 1995). Recently a few researches have been conducted for discrete combinatorial optimization problems. In addition, the applications of *PSO* on combinatorial optimization problems are still considered limited, but the advantages of *PSO* include a simple structure, immediately accessible

for practical applications, ease of implementation, speed to acquire solutions, and robustness that are sustained in the literature.

This study is one of the early applications of *PSO* for discrete combinatorial optimization problems. In addition, to make a confidential comparison both discrete and continuous version of *PSO* is proposed. To remedy the drawback of *PSO* continuous nature, this thesis employed a newly designed method to update particle positions (Pan et al., 2006). Also, the solutions obtained from the metaheuristics methods can be improved by hybridizing the algorithm with local search. Thus, a local search algorithm to neighbors of the global best position vector is proposed for both *CPSO* and *DPSO*.

In conclusion, in this thesis, four *PSO* algorithms namely, *CPSO, DPSO,* $CPSO_{LS}$ and $DPSO_{LS}$ are proposed to solve *UFL* problem. The algorithms have been tested on several benchmark problem instances from OR library and optimal results are obtained in a reasonable computing time. In addition, to the best of our knowledge, these *PSO* algorithms are the first applications of *PSO* reported for the *UFL* in the literature.

For future studies, parallel PSO algorithms can be employed to get more robust results in respect of CPU time. In addition, according to no-free-lunch theorem it is impossible to justify a correlation between reproduction of a training set and generalization error off of the training set using only a priori reasoning. Thus, after tuning the parameters it is more probably that getting more robust results for each algorithm. Also there are a number of local search algorithms but that have been implemented in this study. Lastly, these proposed algorithms can be implemented to other combinatorial optimization problems that have not been tried to solve with PSO.

REFERENCES

Allahverdi, A., Al-Anzi, F.S., "A PSO and a Tabu Search Heuristics for the Assembly Scheduling Problem of the Two-stage Distributed Database Application", *Computers&Operations Research*, Vol. 33, pp. 1056-1080, 2006.

Al-Sultan, K. S., Al-Fawzan, M.A., "A Tabu Search Approach to the Uncapacitated Facility Location Problem", *Annals of Operations Research*, Vol. 86, pp. 91-103, 1999.

Alves, M. L., Almeida, M.T., "Simulated Annealing Algorithm for Simple Plant Location Problems", *Revista Investigacao Operacional*, Vol. 12, pp. 1992.

Aydin, M. E., Fogarty, T.C, "A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimization Problems", *Journal of Heuristics*, Vol. 10, pp. 269-292, 2004.

Barcelo, J., Hallefjord,A., Fernandez, E., Jörnsten, K., "Lagrengean Relaxation and Constraint Generation Procedures for Capacitated Plant Location Problems with Single Sourcing", *O R Spektrum*, Vol. 12, pp. 79-78, 1990.

Beasley, J. E., P.C. Chu, "A Genetic Algorithm For the Set Covering Problem", *European Journal of Operational Research*, Vol. 94, pp. 392–404, 1996.

Beresnev, V., "About one class of problems of parameter optimization of similar technical system", *Manage systems*, Vol. 9, pp. 65-74, 1971.

Beresnev, V. L., Gimady Ed. H., Dementev V. T., "Extremal problems of standartization", *Novosibirsk : Science*, Vol. pp. 1978.

Brandstatter, B., Baumgartner, U., "Particle Swarm Optimization: Mass-spring System Analogon", *IEEE Transactions on Magnetics*, Vol. 38, pp. 997-1000, 2002.

Clerc, M., "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization.", *Proceedings of the Congress on Evolutionary Computation (ICEC99)*, Washington, DC, Vol. pp. 1951-1957, 1999.

Clerc, M., Kennedy, J., "The Particle Swarm Explosion, Stability, and Convergence in a Multidimensional Complex Space", *IEEE Transactions on Evolutionary*

Computation, Vol. 6, pp. 58- 73, 2002.

Conn, A. R., Cornuejols., G., "A Projection Method for the Uncapacitated Facility Location Problem", *Math. Programming*, Vol. 46, pp. 273–298, 1990.

Cornuéjols, G., Fisher, M.L., Nemhauser, G.L., "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms.", *Management Science*, Vol. 22, pp. 789-810, 1977.

Cornuéjols, G., Nemhauser, G.L.,Wolsey, L.A, "The Uncapacitated Facility Location Problem. Discrete Location Theory", *Lecture Note in Artificial Intelligence (LNAI 1865)*, New York, Vol. pp. 119-171, 1990.

Daskin, M. S., *Network and Discrete Location, Models, Algorithms, and Applications.,* pp. Wiley, NewYork, 1995.

Devore, J. L., *Probability and Statistics for Engineering and the Sciences,* pp. Duxbury Thomson Learning, Pacific Grove, CA, USA, 2000.

Eberhart, R. C., Kennedy, J., "A New Optimizer Using Particle Swarm Theory", *Sixth International Symposium on Micro Machine and Human Science*, Nagoya Japan, Vol. pp. 39-43, 1995.

Eberhart, R. C., Simpson, P. K., and Dobbins, R. W., *Compufational Intelligence PC Tools,* pp. Academic Press Professional., Boston, MA:, 1996.

Eberhart, R. C., Shi, Y., "Comparison Between Genetic Algorithms and Particle Swarm Optimization", *Evolutionary programming VII, Lecture Notes in Computer Science*, Porto, Vol. 1447, pp. 611-616, 1998.

Eberhart, R. C., Shi, Y., "Particle Swarm Optimization: Developments, Applications and Resources", *Congress on Evolutionary Computation, IEEE Service Center*, Seoul, Korea, Vol. pp. 81-86, 2001.

Eberhart, R. C., Shi, Y., "Tracking and Optimizing Dynamic Systems with Particle Swarms", *Congress on Evolutionary computation, IEEE Service Center*, Seoul, Korea, Vol. pp. 2001.

Erlenkotter, D., "A Dual-based Procedure for Uncapacitated Facility Location", *Operations Research*, Vol. 26, pp. 992-1009, 1978.

Gen, M., Tsujimura,Y., Ishizaki, S., "Optimal Design of a Star-LAN using Neural Networks", *Computers and Industrial Engineering*, Vol. 31, pp. 855-859, 1996.

Ghosh, D., "Neighborhood Search Heuristics for the Uncapacitated Facility Location Problem", *European Journal of Operation Research*, Vol. pp. 150-162, 2002.

Gimady, E. H., "The Choice of Optimal Scales in One Problem Class of Location or Standardization", *Manage Systems*, Vol. 6, pp. 57-70, 1970.

Guignard, M., "A Lagrangean Dual Ascent Algorithm for Simple Plant Location Problems", *European Journal of Operational Research*, Vol. 35, pp. 193–200, 1988.

Holmberg, K., *Experiments with Primal-Dual Decomposition and Subgradient Methods for the Uncapacitated Facility Location Problem,* Department of Mathematics, Linkoping Institute of Technology,, Research Report LiTH-MAT/OPT-WP-1995-08, 1995.

Holmberg, K. a. K. J., "Dual Search Procedures for The Exact Formulation of The Simple Plant Location Problem with Spatial Interaction", *Location Science*, Vol. 4, pp. 83–100, 1996.

Jaramillo, J. H., Bhadury, J., Batta, R., "On the Use of Genetic Algorithms to Solve Location Problems", *Computers & Operations Research*, Vol. 29, pp. 761-779, 2002.

Kennedy, J., Eberhart, R.C., "Particle Swarm Optimization", *Proceedings of IEEE International Conference on Neural Networks,*, Piscataway, NJ, USA, Vol. pp. 1942-1948, 1995.

Kennedy, J., Eberhart,R.C., Shi.,Y., *Swarm Intelligence,* pp. Morgan Kaufmann, San Mateo, CA, 2001.

Khumawala, B. M., "An Efficient Branch-Bound Algorithm for the Warehouse Location Problem", *Management Science*, Vol. 18, pp. 718-731, 1972.

Klose, A., "A Branch and Bound Algorithm for an UFLP with a Side Constraint", *Int. Trans. Operational Research*, Vol. 5, pp. 155-168, 1998.

Körkel, M., "On the exact solution of large-scale simple plant location problems", *European Journal of Operational Research*, Vol. 39(1), pp. 157–173, 1989.

Korupolu, M. R. a. P., C.G., "Analysis of A Local Search Heuristic For Facility Location Problems", *J. Algorithms*, Vol. 37, pp. 146-188, 2000.

Krarup, J., Pruzan, P.M., "The Simple Plant Location Problem: Survey and Synthesis", *European Journal of Operational Research*, Vol. 12, pp. 36-81, 1983.

Kratica, J., Tosic, D., Filipovic, V., Ljubic, I., "Solving the Simple Plant Location Problems by Genetic Algorithm", *RAIRO Operations Research*, Vol. 35, pp. 127-142, 2001.

Kuehn, A. A., Hamburger, M.J., "A heuristic program for locating warehouses", *Management Science*, Vol. 9, pp. 643–666, 1963.

Laurent, M., Hentenryck, P.V., "A Simple Tabu Search for Warehouse Location", *European J. Oper. Res.*, Vol. 157, pp. 576-591, 2004.

Liao, C. J., Tseng, C-T., Luarn, P., "A Discrete Version of Particle Swarm Optimization

for Flowshop Scheduling Problems", *Computers & Operations Research*, Vol. pp. in press.

Michalewicz, Z., Fogel, D. B., "How to Solve It: Modern Heuristics", *Springer-Verlag*, 2002.

Michel, L., Van Hentenryck, P, "A Simple Tabu Search for Warehouse Location", *European Journal of Operational Research*, Vol. 157, pp. 576-591, 2004.

Mirchandani, P. B., Francis, R.L. (Eds.), "Discrete Location Theory", *Lecture Notes in Artificial Intelligence (LNAI 1865)*, New York, Vol. pp. 1990.

Ozcan, E., Mohan, C.K., "Particle Swarm Optimization: Surfing the Waves", *Proceedings of the Congress on Evolutionary Computation (CEC99)*, Vol. pp. 1939-1944, 1999.

Pan, Q.-K., Tasgetiren, M.F., Liang, Y.-C., "A Discrete Particle Swarm Optimization Algorithm for the No-Wait Flowshop Scheduling Problem", *Ant 2006*, Brussels, Belgium, Vol. pp. 2006.

Parsopoulos, K. E., Vrahatis, M.N., "Recent Approaches to Global Optimization Problems through Particle Swarm Optimization", *Natural Computing*, Vol. 1, pp. 235- 306, 2002.

Rardin , R. L., *Optimization In Operations Research,* pp. Prentice Hall, New Jersey, 1998.

Salman, A., Ahmad, I., Al-Madani, S, "Particle Swarm Optimization for Task Assignment Problem", *Microprocessors and Microsystems*, Vol. 26, pp. 363-371, 2003.

Shi, Y., Eberhart, R.C., "Parameter Selection in Particle Swarm Optimization", *Evolutionary Programming VII: Proceedings of EP 98*, New York, Vol. pp. 591-600, 1998.

Shi, Y., Eberhart, R.C., "A Modified Particle Swarm Optimizer", *Proceedings ofthe IEEE International Conference on Evolutionary Computation*, Piscataway, NJ, Vol. pp. 303-308, 1998.

Shi, Y., Eberhart, R.C., "Empirical Study of Particle Swarm Optimization,", *Proceedings of the Congress on Evolutionary Computation (CEC99)*, Vol. pp. 1945-1950, 1999.

Shigenori, N., Takamu, G., Toshiku, Y., Yoshikazu, F, "A hybrid particle swarm optimization for distribution state estimation", *IEEE Trans. on Power Sys.*, Vol. 18, pp. 60-68, 2003.

Simao, H. P., J.M., Thizy, "A Dual Simplex Algorithm for the Canonical Representation of the Uncapacitated

Facility Location Problem", *Operations Research Letters*, Vol. 8, pp. 279–286, 1989.

Simchi-Levi, D., Kaminsky, P, Simchi-Levi, E., *Designing and managing the supply chain, concepts, strategies and case studies.,* pp. Irwin McGraw-Hill, Boston, MA, 2000.

Sugantha, P. N., "Particle Swarm Optimization with Neighborhood Operator", *Proceedings of the Congress on Evolutionary Computation (CEC99)*, Vol. pp. 1958-1962., 1999.

Sun, M., "Solving the Uncapacitated Facility Location Problem Using Tabu Search", *Computers & Operations Research*, Vol. 33, pp. 2563–2589, 2006.

Tandon, V., *Closing the Gap Between CAD/CAM and Optimized CNC End Milling.,* Master, Indiana Un. Purdue Un., Purdue School of Eng. and Tech., 2000.

Tasgetiren, M. F., Liang, Y.-C., "A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem"*, Journal of Economic and Social Research*, Vol. 5, pp. 1-20, 2003.

Trelea, I. C., "The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection", *Information Processing Letters*, Vol. 85, pp. 317- 325, 2003.

Vaithyanathan, S., Burke, L., Magent, M, "Massively Parallel Analog Tabu Search Using Neural Networks Applied to Simple Plant Location Problems", *European Journal of Operational Research*, Vol. 93, pp. 317-330, 1996.

Van den Bergh, F., Engelbecht, A. P., "Cooperative Learning in Neural Networks Using Particle Swarm Optimizers", *South African Computer Journal*, Vol. 26, pp. 84-90, 2000.

Van Roy, T. J., "A Cross Decomposition Algorithm for Capacitated Facility Location,", *Operations Research*, Vol. 34, pp. 145-163, 1986.

Yeh, L. W., *Optimal Procurement Policies for Multi-Product Multi-Supplier with Capacity Constraint and Price Discount,* Master, Yuan Ze University, Department of Industrial Engineering and Management, 2003.

Yin, P. Y., "A Discrete Particle Swarm Algorithm for Optimal Polygonal Approximation of Digital Curves", *J. Visual Comm. and Image Repr*, Vol. 15, pp. 241-260, 2004.

YinT, P.-Y., Yu,S-S., Wang, P-P., Wang,Y-T., "A Hybrid Particle Swarm Optimization Algorithm for Optimal Task Assignment in Distributed Systems", *Computer Standards & Interfaces*, 2005.

Yoshida, H., Kawata, K., Fukuyama, Y. and Nakanishi, Y., "A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment", *IEEE Transactions on Power Systems*, Vol. 15, pp. 1232-1239, 2000.

APPENDIX A

EXPERIMENTAL RESULTS FOR CPSO

Table A.1 Experimental Results of Fitness and CPU for CPSO (Cap71-74, Cap81-82)

Problem	Cap71	Cap72	Cap73	Cap74	Cap81	Cap82
Size	16×50	16×50	16×50	16×50	25*50	25*50
Optimum	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R1	934199.14	977799.40	1010808.16	1034976.98	800004.98	856040.14
R2	933568.90	977799.40	1010641.45	1034976.98	800815.63	857380.85
R3	932615.75	977799.40	1010641.45	1034976.98	802457.23	857307.69
R4	932615.75	979099.61	1010641.45	1034976.98	797508.73	857307.69
R5	932615.75	977799.40	1010641.45	1034976.98	797508.73	856879.16
R6	934199.14	977799.40	1012476.98	1034976.98	797582.29	857271.96
R7	932615.75	977799.40	1010641.45	1034976.98	797508.73	855466.85
R8	932615.75	977799.40	1010641.45	1034976.98	799695.03	856767.06
R9	932615.75	977799.40	1010641.45	1034976.98	797508.73	857307.69
R10	932615.75	983713.81	1010641.45	1034976.98	797508.73	854704.20
R11	932615.75	977799.40	1010641.45	1037717.08	797508.73	855466.85
R12	932615.75	979099.61	1010641.45	1045342.23	797508.73	854704.20
R13	932615.75	977799.40	1012476.98	1034976.98	798319.38	854704.20
R14	932615.75	977799.40	1010641.45	1034976.98	798243.31	855466.85
R15	932615.75	977799.40	1010641.45	1037717.08	800958.13	856287.59
R16	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R17	932615.75	977799.40	1012476.98	1037717.08	797508.73	855971.75
R18	932615.75	980176.51	1010641.45	1034976.98	797508.73	855971.75
R19	932615.75	977799.40	1011067.65	1034976.98	796648.44	855781.10
R20	932615.75	977799.40	1010641.45	1034976.98	796648.44	856287.59
R21	932615.75	977799.40	1010641.45	1037717.08	797508.73	854704.20
R22	932615.75	977799.40	1011067.65	1034976.98	796648.44	854704.20
R23	932615.75	977799.40	1010641.45	1034976.98	797508.73	856004.41
R24	932615.75	977799.40	1010641.45	1037717.08	797508.73	854704.20
R25	932615.75	977799.40	1010641.45	1037717.08	800153.53	854704.20
R26	934199.14	977799.40	1010641.45	1037717.08	797508.73	854704.20
R27	932615.75	981538.85	1010641.45	1034976.98	796648.44	855466.85
R28	932615.75	977799.40	1012643.69	1034976.98	798325.94	854704.20
R29	932615.75	977799.40	1010641.45	1034976.98	796648.44	856879.16
R30	934199.14	977799.40	1012476.98	1034976.98	799144.69	857271.96
Best Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Worse Fitness	934199.14	983713.81	1012643.69	1045342.23	802457.23	857380.85
Std Dev. of Fitness	562.23	1324.30	702.13	2124.54	1480.72	1015.64
ARPE	0.03	0.05	0.03	0.10	0.18	0.13
HR	0.83	0.83	0.73	0.00	0.00	0.33
Average CPU	0.1318	0.1318	0.1865	0.1781	0.8823	0.7672
Minimum CPU	0.0150	0.0000	0.0000	0.0000	0.0470	0.0620
Maximum CPU	0.6250	0.6100	0.5940	0.5940	1.1100	1.1100
Std Dev. of CPU	0.2131	0.2159	0.2456	0.2424	0.3928	0.4395

Table A.2 Experimental Results of Fitness and CPU for CPSO (Cap83-84, Cap91-94)

Problem	Cap83	Cap84	Cap91	Cap92	Cap93	Cap94
Size	25*50	25*50	25*50	25*50	25*50	25*50
Optimum	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R1	894008.14	934586.98	800004.98	856040.14	894008.14	934586.98
R2	894801.16	928941.75	800815.63	857380.85	894801.16	928941.75
R3	895049.66	928941.75	802457.23	857307.69	895049.66	928941.75
R4	893782.11	934586.98	797508.73	857307.69	893782.11	934586.98
R5	894801.16	928941.75	797508.73	856879.16	894801.16	928941.75
R6	894801.16	938912.54	797582.29	857271.96	894801.16	938912.54
R7	893782.11	944394.83	797508.73	855466.85	893782.11	944394.83
R8	894801.16	928941.75	799695.03	856767.06	894801.16	928941.75
R9	899348.08	928941.75	797508.73	857307.69	899348.08	928941.75
R10	894008.14	928941.75	797508.73	854704.20	894008.14	928941.75
R11	894008.14	928941.75	797508.73	855466.85	894008.14	928941.75
R12	894008.14	928941.75	797508.73	854704.20	894008.14	928941.75
R13	894008.14	934586.98	798319.38	854704.20	894008.14	934586.98
R14	893782.11	928941.75	798243.31	855466.85	893782.11	928941.75
R15	893782.11	935106.20	800958.13	856287.59	893782.11	935106.20
R16	895027.19	928941.75	796648.44	854704.20	895027.19	928941.75
R17	898800.04	928941.75	797508.73	855971.75	898800.04	928941.75
R18	894008.14	928941.75	797508.73	855971.75	894008.14	928941.75
R19	893782.11	934586.98	796648.44	855781.10	893782.11	934586.98
R20	895027.19	934586.98	796648.44	856287.59	895027.19	934586.98
R21	895027.19	934586.98	797508.73	854704.20	895027.19	934586.98
R22	894008.14	934586.98	796648.44	854704.20	894008.14	934586.98
R23	894008.14	928941.75	797508.73	856004.41	894008.14	928941.75
R24	894573.71	928941.75	797508.73	854704.20	894573.71	928941.75
R25	894008.14	928941.75	800153.53	854704.20	894008.14	928941.75
R26	895027.19	928941.75	797508.73	854704.20	895027.19	928941.75
R27	893782.11	934586.98	796648.44	855466.85	893782.11	934586.98
R28	894008.14	932007.96	798325.94	854704.20	894008.14	932007.96
R29	894008.14	928941.75	796648.44	856879.16	894008.14	928941.75
R30	895027.19	928941.75	799144.69	857271.96	895027.19	928941.75
Best Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Worse Fitness	899348.08	944394.83	802457.23	857380.85	899348.08	944394.83
Std Dev. of Fitness	1303.87	3842.64	1480.72	1015.64	1303.87	3842.64
ARPE	0.09	0.29	0.18	0.13	0.09	0.29
HR	0.00	0.60	0.00	0.33	0.00	0.60
Average CPU	1.2474	0.6000	0.8813	0.7667	1.2516	0.6021
Minimum CPU	1.0620	0.0460	0.0470	0.0620	1.0470	0.0460
Maximum CPU	1.3910	1.3280	1.1090	1.1250	1.3910	1.3290
Std Dev. of CPU	0.1435	0.5869	0.3946	0.4422	0.1464	0.5882

Table A.3 Experimental Results of Fitness and CPU for CPSO (Cap101-104, Cap111-112)

Problem	Cap101	Cap102	Cap103	Cap104	Cap111	Cap112
Size	25×50	25×50	25×50	25×50	50*50	50*50
Optimum	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R1	800004.98	856040.14	894008.14	934586.98	802393.25	858396.93
R2	800815.63	857380.85	894801.16	928941.75	803223.43	852762.88
R3	802457.23	857307.69	895049.66	928941.75	799900.45	852326.39
R4	797508.73	857307.69	893782.11	934586.98	800057.50	854823.78
R5	797508.73	856879.16	894008.14	928941.75	802754.66	860161.76
R6	797582.29	857271.96	894008.14	938912.54	799336.94	860558.90
R7	797508.73	855466.85	894801.16	944394.83	800247.98	861949.05
R8	799695.03	856767.06	893782.11	928941.75	798501.61	856543.75
R9	797508.73	857307.69	894801.16	928941.75	802541.51	856717.93
R10	797508.73	854704.20	899424.91	928941.75	804205.46	859873.49
R11	797508.73	855466.85	895275.69	928941.75	804432.09	852515.26
R12	797508.73	854704.20	895027.19	928941.75	801786.78	864584.21
R13	798319.38	854704.20	894008.14	934586.98	801793.94	854998.58
R14	798243.31	855466.85	895027.19	928941.75	804549.64	868567.16
R15	800958.13	856287.59	894801.16	935106.20	801731.16	854450.29
R16	796648.44	854704.20	899122.05	928941.75	803552.28	856082.48
R17	797508.73	855971.75	899424.91	928941.75	796005.36	861468.01
R18	797508.73	855971.75	894801.16	928941.75	801299.53	854704.20
R19	796648.44	855781.10	893782.11	934586.98	800353.91	855768.21
R20	796648.44	856287.59	898500.95	934586.98	796787.98	859291.35
R21	797508.73	854704.20	893782.11	934586.98	799874.94	860849.81
R22	796648.44	854704.20	894008.14	934586.98	800418.73	851495.33
R23	797508.73	856004.41	895027.19	928941.75	795291.86	865035.73
R24	797508.73	854704.20	894801.16	928941.75	801538.95	852762.88
R25	800153.53	854704.20	894008.14	928941.75	797452.36	858381.63
R26	797508.73	854704.20	894801.16	928941.75	799035.75	864603.66
R27	796648.44	855466.85	893782.11	934586.98	800131.54	854240.36
R28	798325.94	854704.20	895027.19	932007.96	802161.86	856679.24
R29	796648.44	856879.16	895027.19	928941.75	801012.90	860089.81
R30	799144.69	857271.96	893782.11	928941.75	797635.29	857396.56
Best Fitness	796648.44	854704.20	893782.11	928941.75	795291.86	851495.33
Worse Fitness	802457.23	857380.85	899424.91	944394.83	804549.64	868567.16
Std Dev. of Fitness	1480.72	1015.64	1695.79	3842.64	2429.54	4297.07
ARPE	0.18	0.13	0.14	0.29	0.91	0.76
HR	0.00	0.33	0.00	0.60	0.00	0.00
Average CPU	0.8818	0.7667	0.9938	0.6026	3.6182	3.5474
Minimum CPU	0.0620	0.0460	0.0310	0.0460	3.4530	0.2180
Maximum CPU	1.1090	1.1250	1.3910	1.3280	3.9380	4.1250
Std Dev. of CPU	0.3910	0.4426	0.4807	0.5866	0.1376	0.6574

Table A.4 Experimental Results of Fitness and CPU for CPSO (Cap113-114, Cap121-124)

Problem	Cap113	Cap114	Cap121	Cap122	Cap123	Cap124
Size	50*50	50*50	50*50	50*50	50*50	50*50
Optimum	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R1	894008.14	934586.98	802393.25	858396.93	894008.14	934586.98
R2	894801.16	944008.53	803223.43	852762.88	894801.16	944008.53
R3	893782.11	928941.75	799900.45	852326.39	893782.11	928941.75
R4	903014.45	929477.56	800057.50	854823.78	903014.45	929477.56
R5	895109.53	951073.38	802754.66	860161.76	895109.53	951073.38
R6	898158.56	928941.75	799336.94	860558.90	898158.56	928941.75
R7	893956.91	940358.03	800247.98	861949.05	893956.91	940358.03
R8	893956.91	929477.56	798501.61	856543.75	893956.91	929477.56
R9	909822.04	935122.79	802541.51	856717.93	909822.04	935122.79
R10	899348.08	928941.75	804205.46	859873.49	899348.08	928941.75
R11	909908.70	932007.96	804432.09	852515.26	909908.70	932007.96
R12	893251.51	928941.75	801786.78	864584.21	893251.51	928941.75
R13	894095.76	931507.55	801793.94	854998.58	894095.76	931507.55
R14	893076.71	938630.55	804549.64	868567.16	893076.71	938630.55
R15	897366.96	937152.78	801731.16	854450.29	897366.96	937152.78
R16	894095.76	944281.61	803552.28	856082.48	894095.76	944281.61
R17	895683.45	941793.78	796005.36	861468.01	895683.45	941793.78
R18	895656.43	939182.38	801299.53	854704.20	895656.43	939182.38
R19	899728.81	934586.98	800353.91	855768.21	899728.81	934586.98
R20	898096.88	937152.78	796787.98	859291.35	898096.88	937152.78
R21	893076.71	939756.90	799874.94	860849.81	893076.71	939756.90
R22	897366.96	928941.75	800418.73	851495.33	897366.96	928941.75
R23	898980.31	929477.56	795291.86	865035.73	898980.31	929477.56
R24	903681.13	928941.75	801538.95	852762.88	903681.13	928941.75
R25	897254.54	939756.90	797452.36	858381.63	897254.54	939756.90
R26	901570.26	929477.56	799035.75	864603.66	901570.26	929477.56
R27	896347.91	951803.25	800131.54	854240.36	896347.91	951803.25
R28	900870.59	928941.75	802161.86	856679.24	900870.59	928941.75
R29	895407.93	932543.78	801012.90	860089.81	895407.93	932543.78
R30	907021.00	935122.79	797635.29	857396.56	907021.00	935122.79
Best Fitness	893076.71	928941.75	795291.86	851495.33	893076.71	928941.75
Worse Fitness	909908.70	951803.25	804549.64	868567.16	909908.70	951803.25
Std Dev. of Fitness	4753.48	6619.05	2429.54	4297.07	4753.48	6619.05
ARPE	0.55	0.69	0.91	0.76	0.55	0.69
HR	0.00	0.23	0.00	0.00	0.00	0.23
Average CPU	3.9031	3.3375	3.6141	3.5495	3.8901	3.3359
Minimum CPU	3.4540	0.1560	3.4220	0.2180	3.4530	0.1410
Maximum CPU	4.5780	4.4840	3.9220	4.1250	4.5620	4.4690
Std Dev. of CPU	0.2892	1.7163	0.1385	0.6577	0.2873	1.7223

Table A.5 Experimental Results of Fitness and CPU for CPSO (Cap131-134)

Problem	Cap131	Cap132	Cap133	Cap134
Size	50×50	50×50	50×50	50×50
Optimum	793439.56	851495.33	893076.71	928941.75
R1	802393.25	858396.93	894008.14	934586.98
R2	803223.43	852762.88	894801.16	944008.53
R3	799900.45	852326.39	893782.11	928941.75
R4	800057.50	854823.78	903014.45	929477.56
R5	802754.66	860161.76	895109.53	951073.38
R6	799336.94	860558.90	898158.56	928941.75
R7	800247.98	861949.05	893956.91	940358.03
R8	798501.61	856543.75	893956.91	929477.56
R9	802541.51	856717.93	909822.04	935122.79
R10	804205.46	859873.49	899348.08	928941.75
R11	804432.09	852515.26	909908.70	932007.96
R12	801786.78	864584.21	893251.51	928941.75
R13	801793.94	854998.58	894095.76	931507.55
R14	804549.64	868567.16	893076.71	938630.55
R15	801731.16	854450.29	897548.76	937152.78
R16	803552.28	856082.48	898580.66	944281.61
R17	796005.36	861468.01	894664.40	941793.78
R18	801299.53	854704.20	898155.53	939182.38
R19	800353.91	855768.21	897020.63	934586.98
R20	796787.98	859291.35	896661.56	937152.78
R21	799874.94	860849.81	899270.31	939756.90
R22	800418.73	851495.33	897270.91	928941.75
R23	795291.86	865035.73	899908.13	929477.56
R24	801538.95	852762.88	897786.11	928941.75
R25	797452.36	858381.63	895407.93	939756.90
R26	799035.75	864603.66	899460.98	929477.56
R27	800131.54	854240.36	894388.88	951803.25
R28	802161.86	856679.24	901392.48	928941.75
R29	801012.90	860089.81	896573.94	932543.78
R30	797635.29	857396.56	894801.16	935122.79
Best Fitness	795291.86	851495.33	893076.71	928941.75
Worse Fitness	804549.64	868567.16	909908.70	951803.25
Std Dev. of Fitness	2429.54	4297.07	4210.93	6619.05
ARPE	0.91	0.76	0.50	0.69
HR	0.00	0.00	0.00	0.23
Average CPU	3.6156	3.5599	3.7792	3.3333
Minimum CPU	3.4220	0.2040	0.3750	0.1560
Maximum CPU	3.9380	4.0940	4.5620	4.4530
Std Dev. of CPU	0.1401	0.6593	0.7184	1.7151

Table A.6 Experimental Results of Fitness and CPU for CPSO (CapA, CapB, CapC)

Problem	CapA	CapB	CapC
Size	100×1000	100×1000	100×1000
Optimum	17156454.48	12979071.58	11505594.33
R1	19286308.21	13535957.21	12482849.89
R2	20642958.19	14183783.00	13572876.10
R3	23726365.27	13915717.03	12666392.80
R4	20273318.69	14391434.98	12625587.01
R5	20131002.97	14377328.68	12010668.40
R6	19560413.25	15225051.19	12432081.74
R7	19184330.20	14490923.59	12194724.73
R8	19435176.01	14317150.70	12601149.73
R9	22448799.61	13829452.17	11973777.65
R10	18351465.40	15356618.36	12156591.03
R11	21173914.15	13694793.35	12647749.55
R12	21511443.49	14184944.23	12608131.71
R13	20618944.62	15180831.87	12292171.01
R14	20628934.73	14270178.25	12830934.37
R15	20215367.64	13774673.75	12709388.98
R16	21022778.19	14137931.84	12632163.92
R17	24436551.73	14129302.76	11982702.88
R18	22258480.09	13416600.21	12147427.35
R19	20569098.99	13864086.57	12258983.32
R20	18481759.03	14557530.50	12936963.15
R21	22349042.07	14535732.12	12452709.10
R22	22068847.13	13390061.37	12604244.96
R23	19603121.97	15179670.38	12115556.52
R24	20246010.78	13864395.24	12137809.45
R25	19644712.77	15066399.48	12446321.24
R26	20743013.10	14580044.76	11970113.17
R27	21455674.11	14020107.96	12816627.25
R28	20654071.91	14332296.27	12549200.10
R29	24638983.70	14598361.54	12153710.70
R30	18664032.38	14435163.65	12330307.11
Best Fitness	18351465.40	13390061.37	11970113.17
Worse Fitness	24638983.70	15356618.36	13572876.10
Std Dev. of Fitness	1608650.32	532161.62	350412.47
ARPE	21.24	10.14	8.16
HR	0.00	0.00	0.00
Average CPU	19.5740	17.1318	17.6149
Minimum CPU	14.5797	14.5453	15.0766
Maximum CPU	26.0891	22.4030	21.3422
Std Dev. of CPU	2.9248	2.0176	1.3452

APPENDIX B

EXPERIMENTAL RESULTS FOR DPSO

Table B.1 Experimental Results of Fitness and CPU for DPSO (Cap71-74, Cap81-82)

Problem	Cap71	Cap72	Cap73	Cap74	Cap81	Cap82
Size	16×50	16×50	16×50	16×50	25*50	25*50
Optimum	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R1	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R2	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R3	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R4	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R5	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R6	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R7	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R8	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R9	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R10	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R11	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R12	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R13	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R14	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R15	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R16	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R17	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R18	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R19	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R20	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R21	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R22	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R23	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R24	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R25	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R26	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R27	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R28	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R29	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R30	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Best Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Worse Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.0641	0.0651	0.0708	0.0693	0.3130	0.3062
Minimum CPU	0.0160	0.0150	0.0150	0.0160	0.1570	0.1720
Maximum CPU	0.1100	0.1100	0.1250	0.1250	0.5940	0.5310
Std Dev. of CPU	0.0263	0.0252	0.0295	0.0294	0.1013	0.0890

Table B.2 Experimental Results of Fitness and CPU for DPSO (Cap83-84, Cap91-94)

Problem	Cap83	Cap84	Cap91	Cap92	Cap93	Cap94
Size	25*50	25*50	25*50	25*50	25*50	25*50
Optimum	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R1	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R2	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R3	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R4	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R5	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R6	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R7	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R8	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R9	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R10	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R11	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R12	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R13	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R14	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R15	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R16	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R17	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R18	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R19	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R20	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R21	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R22	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R23	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R24	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R25	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R26	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R27	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R28	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R29	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R30	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Best Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Worse Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.9365	0.2011	0.3119	0.3063	0.9333	0.2016
Minimum CPU	0.9060	0.1090	0.1560	0.1560	0.8910	0.1100
Maximum CPU	0.9540	0.3280	0.5940	0.5320	0.9680	0.3130
Std Dev. of CPU	0.0142	0.0629	0.1022	0.0887	0.0173	0.0628

Table B.3 Experimental Results of Fitness and CPU for DPSO (Cap101-104, Cap111-112)

Problem	Cap101	Cap102	Cap103	Cap104	Cap111	Cap112
Size	25×50	25×50	25×50	25×50	50*50	50*50
Optimum	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R1	796648.44	854704.20	893782.11	928941.75	795927.69	854061.12
R2	796648.44	854704.20	893782.11	928941.75	794299.85	852151.59
R3	796648.44	854704.20	893782.11	928941.75	795034.44	853006.06
R4	796648.44	854704.20	893782.11	928941.75	794159.35	851495.33
R5	796648.44	854704.20	893782.11	928941.75	794299.85	851670.12
R6	796648.44	854704.20	893782.11	928941.75	794299.85	851670.12
R7	796648.44	854704.20	894008.14	928941.75	795291.86	851495.33
R8	796648.44	854704.20	893782.11	928941.75	795291.86	852864.74
R9	796648.44	854704.20	893782.11	928941.75	795291.86	852572.23
R10	796648.44	854704.20	893782.11	928941.75	794299.85	851670.12
R11	796648.44	854704.20	893782.11	928941.75	795614.98	851670.12
R12	796648.44	854704.20	893782.11	928941.75	795708.95	852257.98
R13	796648.44	854704.20	893782.11	928941.75	794299.85	852690.06
R14	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R15	796648.44	854704.20	893782.11	928941.75	794448.40	852904.43
R16	796648.44	854704.20	893782.11	928941.75	794299.85	853487.53
R17	796648.44	854704.20	893782.11	928941.75	793439.56	852151.59
R18	796648.44	854704.20	893782.11	928941.75	794299.85	852432.78
R19	796648.44	854704.20	893782.11	928941.75	794956.11	853171.53
R20	796648.44	854704.20	893782.11	928941.75	795836.85	852151.59
R21	796648.44	854704.20	893782.11	928941.75	795110.50	851670.12
R22	796648.44	854704.20	893782.11	928941.75	793439.56	852690.06
R23	796648.44	854704.20	893782.11	928941.75	795614.98	852151.59
R24	796648.44	854704.20	893782.11	928941.75	795110.50	851670.12
R25	796648.44	854704.20	893782.11	928941.75	794299.85	851495.33
R26	796648.44	854704.20	893782.11	928941.75	794299.85	852572.23
R27	796648.44	854704.20	894008.14	928941.75	793439.56	851670.12
R28	796648.44	854704.20	893782.11	928941.75	794956.11	852257.98
R29	796648.44	854704.20	893782.11	928941.75	795883.24	853039.54
R30	796648.44	854704.20	893782.11	928941.75	794956.11	851495.33
Best Fitness	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Worse Fitness	796648.44	854704.20	894008.14	928941.75	795927.69	854061.12
Std Dev. of Fitness	0.00	0.00	57.35	0.00	759.45	683.00
ARPE	0.00	0.00	0.00	0.00	0.16	0.09
HR	1.00	1.00	0.93	1.00	0.13	0.17
Average CPU	0.3130	0.3062	0.3625	0.2021	2.6141	2.6589
Minimum CPU	0.1570	0.1570	0.0930	0.1090	2.5780	2.0000
Maximum CPU	0.5940	0.5310	0.9540	0.3130	2.6570	3.2970
Std Dev. of CPU	0.1006	0.0915	0.1960	0.0627	0.0198	0.2142

Table B.4 Experimental Results of Fitness and CPU for DPSO (Cap113-114, Cap121-124)

Problem	Cap113	Cap114	Cap121	Cap122	Cap123	Cap124
Size	50*50	50*50	50*50	50*50	50*50	50*50
Optimum	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R1	893076.71	928941.75	795927.69	854061.12	893076.71	928941.75
R2	893907.77	928941.75	794299.85	852151.59	893907.77	928941.75
R3	893076.71	928941.75	795034.44	853006.06	893076.71	928941.75
R4	893076.71	928941.75	794159.35	851495.33	893076.71	928941.75
R5	893907.77	928941.75	794299.85	851670.12	893907.77	928941.75
R6	893782.11	928941.75	794299.85	851670.12	893782.11	928941.75
R7	893251.51	928941.75	795291.86	851495.33	893251.51	928941.75
R8	893076.71	928941.75	795291.86	852864.74	893076.71	928941.75
R9	894008.14	928941.75	795291.86	852572.23	894008.14	928941.75
R10	893076.71	928941.75	794299.85	851670.12	893076.71	928941.75
R11	894008.14	928941.75	795614.98	851670.12	894008.14	928941.75
R12	893076.71	928941.75	795708.95	852257.98	893076.71	928941.75
R13	894095.76	928941.75	794299.85	852690.06	894095.76	928941.75
R14	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R15	894008.14	928941.75	794448.40	852904.43	894008.14	928941.75
R16	893076.71	928941.75	794299.85	853487.53	893076.71	928941.75
R17	893076.71	928941.75	793439.56	852151.59	893076.71	928941.75
R18	893076.71	928941.75	795935.81	852432.78	893076.71	928941.75
R19	893076.71	928941.75	794956.11	853171.53	893076.71	928941.75
R20	893782.11	928941.75	795836.85	852151.59	893782.11	928941.75
R21	893732.98	928941.75	795110.50	851670.12	893732.98	928941.75
R22	893782.11	928941.75	793439.56	852690.06	893782.11	928941.75
R23	893076.71	928941.75	795883.24	852151.59	893076.71	928941.75
R24	894095.76	928941.75	795110.50	851670.12	894095.76	928941.75
R25	893076.71	928941.75	794299.85	851495.33	893076.71	928941.75
R26	893907.77	928941.75	794299.85	852572.23	893907.77	928941.75
R27	893907.77	928941.75	793439.56	851670.12	893907.77	928941.75
R28	893076.71	928941.75	795883.24	852257.98	893076.71	928941.75
R29	893251.51	928941.75	795883.24	853039.54	893251.51	928941.75
R30	893782.11	928941.75	794956.11	851495.33	893782.11	928941.75
Best Fitness	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Worse Fitness	894095.76	928941.75	795935.81	854061.12	894095.76	928941.75
Std Dev. of Fitness	420.88	0.00	822.12	683.00	420.88	0.00
ARPE	0.04	0.00	0.17	0.09	0.04	0.00
HR	0.47	1.00	0.13	0.17	0.47	1.00
Average CPU	2.7984	1.7136	2.5453	2.6318	2.8167	1.7146
Minimum CPU	2.7350	1.0930	1.8590	1.9840	2.7340	1.0940
Maximum CPU	2.9370	2.3440	2.6560	2.7190	3.2820	2.3430
Std Dev. of CPU	0.0416	0.3225	0.1804	0.1758	0.0990	0.3239

Table B.5 Experimental Results of Fitness and CPU for DPSO (Cap131-134)

Problem	Cap131	Cap132	Cap133	Cap134
Size	50×50	50×50	50×50	50×50
Optimum	793439.56	851495.33	893076.71	928941.75
R1	795927.69	854061.12	893076.71	928941.75
R2	794299.85	852151.59	893907.77	928941.75
R3	795034.44	853006.06	893076.71	928941.75
R4	794159.35	851495.33	893732.98	928941.75
R5	794299.85	851670.12	893907.77	928941.75
R6	794299.85	851670.12	893782.11	928941.75
R7	795291.86	851495.33	893251.51	928941.75
R8	795291.86	852864.74	893076.71	928941.75
R9	795291.86	852572.23	893251.51	928941.75
R10	794299.85	851670.12	893251.51	928941.75
R11	795614.98	851670.12	894008.14	928941.75
R12	795708.95	852257.98	893076.71	928941.75
R13	794299.85	852690.06	893076.71	928941.75
R14	793439.56	851495.33	893076.71	928941.75
R15	794448.40	852904.43	893076.71	928941.75
R16	794299.85	853487.53	893076.71	928941.75
R17	793439.56	852151.59	893076.71	928941.75
R18	795935.81	852432.78	894008.14	928941.75
R19	794956.11	853171.53	893076.71	928941.75
R20	795836.85	852151.59	893956.91	928941.75
R21	795110.50	851670.12	893732.98	928941.75
R22	793439.56	852690.06	893782.11	928941.75
R23	795883.24	852151.59	893076.71	928941.75
R24	795110.50	851670.12	894095.76	928941.75
R25	794299.85	851495.33	893076.71	928941.75
R26	794299.85	852572.23	893907.77	928941.75
R27	793439.56	851670.12	893907.77	928941.75
R28	795883.24	852257.98	893076.71	928941.75
R29	795883.24	853039.54	893251.51	928941.75
R30	794956.11	851495.33	893782.11	928941.75
Best Fitness	793439.56	851495.33	893076.71	928941.75
Worse Fitness	795935.81	854061.12	894095.76	928941.75
Std Dev. of Fitness	822.12	683.00	398.07	0.00
ARPE	0.17	0.09	0.04	0.00
HR	0.13	0.17	0.43	1.00
Average CPU	2.5464	2.6328	2.5292	1.7167
Minimum CPU	1.8590	1.9850	1.3120	1.0940
Maximum CPU	2.6570	2.7340	2.8440	2.3440
Std Dev. of CPU	0.1802	0.1765	0.4015	0.3236

Table B.6 Experimental Results of Fitness and CPU for DPSO (CapA, CapB, CapC)

Problem	CapA	CapB	CapC
Size	100×1000	100×1000	100×1000
Optimum	17156454.48	12979071.58	11505594.33
R1	18443018.79	13531212.89	11962452.14
R2	18894956.98	13631186.37	11925228.95
R3	18891645.29	13543359.13	12017221.75
R4	18187486.86	13642369.37	11978021.05
R5	18917956.67	13404369.23	12062754.08
R6	19060412.50	13560684.88	11906158.35
R7	18005287.27	13683285.12	12002066.48
R8	18352511.86	13752850.49	12068311.47
R9	18358038.12	13287703.69	12045401.22
R10	18797093.73	13714841.04	12120511.41
R11	19085419.83	13695957.63	11997640.45
R12	18905525.98	13612933.66	12180785.60
R13	18819343.93	13636670.96	11966273.49
R14	17810901.35	13675479.60	12136713.60
R15	18206354.48	13567406.09	12116639.78
R16	18454922.80	13307568.34	12002874.65
R17	18745685.48	13694580.17	12122326.90
R18	18503515.88	13635318.61	11855107.73
R19	18750964.16	13717310.63	12131028.17
R20	18837421.82	13499552.61	12177213.75
R21	18772381.74	13681660.29	11980596.86
R22	18033096.90	13923731.30	12143949.64
R23	18974860.60	13611033.23	12053339.47
R24	18368475.19	13480454.01	11939968.78
R25	18919372.58	13802226.64	12018782.14
R26	18729636.32	13512324.14	11948467.25
R27	18354003.11	13638769.69	12028610.32
R28	19028995.87	13663028.65	11930936.11
R29	18658660.67	13771439.04	11958433.47
R30	19366467.31	13641296.69	12076767.10
Best Fitness	17810901.35	13287703.69	11855107.73
Worse Fitness	19366467.31	13923731.30	12180785.60
Std Dev. of Fitness	367959.31	135262.96	85472.59
ARPE	8.65	4.92	4.54
HR	0.00	0.00	0.00
Average CPU	17.8973	17.0653	17.1341
Minimum CPU	17.7171	16.9203	17.0156
Maximum CPU	18.1500	17.1735	17.3250
Std Dev. of CPU	0.0896	0.0636	0.0720

APPENDIX C

EXPERIMENTAL RESULTS FOR $CPSO_{LS}$

Table C.1 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap71-74, Cap81-82)

Problem	Cap71	Cap72	Cap73	Cap74	Cap81	Cap82
Size	16×50	16×50	16×50	16×50	25*50	25*50
Optimum	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R1	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R2	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R3	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R4	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R5	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R6	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R7	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R8	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R9	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R10	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R11	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R12	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R13	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R14	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R15	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R16	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R17	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R18	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R19	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R20	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R21	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R22	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R23	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R24	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R25	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R26	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R27	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R28	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R29	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R30	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Best Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Worse Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.0146	0.0172	0.0281	0.0182	0.1875	0.0896
Minimum CPU	0.0000	0.0000	0.0000	0.0000	0.0000	0.0150
Maximum CPU	0.0630	0.0780	0.0930	0.0630	0.7650	0.2970
Std Dev. of CPU	0.0164	0.0207	0.0250	0.0190	0.1938	0.0750

Table C.2 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap83-84, Cap91-94)

Problem	Cap83	Cap84	Cap91	Cap92	Cap93	Cap94
Size	25*50	25*50	25*50	25*50	25*50	25*50
Optimum	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R1	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R2	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R3	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R4	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R5	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R6	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R7	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R8	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R9	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R10	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R11	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R12	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R13	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R14	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R15	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R16	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R17	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R18	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R19	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R20	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R21	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R22	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R23	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R24	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R25	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R26	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R27	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R28	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R29	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R30	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Best Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Worse Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.2151	0.0370	0.1875	0.0896	0.2151	0.0370
Minimum CPU	0.0310	0.0000	0.0000	0.0000	0.0310	0.0000
Maximum CPU	0.7500	0.1880	0.7660	0.2970	0.7500	0.1720
Std Dev. of CPU	0.1955	0.0393	0.1939	0.0756	0.1964	0.0399

Table C.3 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap101-104, Cap111-112)

Problem	Cap101	Cap102	Cap103	Cap104	Cap111	Cap112
Size	25×50	25×50	25×50	25×50	50*50	50*50
Optimum	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R1	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R2	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R3	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R4	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R5	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R6	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R7	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R8	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R9	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R10	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R11	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R12	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R13	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R14	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R15	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R16	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R17	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R18	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R19	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R20	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R21	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R22	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R23	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R24	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R25	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R26	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R27	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R28	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R29	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R30	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Best Fitness	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Worse Fitness	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.1880	0.0906	0.2151	0.0370	1.4271	1.0245
Minimum CPU	0.0000	0.0000	0.0310	0.0000	0.0780	0.1090
Maximum CPU	0.7650	0.2970	0.7500	0.1720	3.4850	4.9370
Std Dev. of CPU	0.1950	0.0774	0.1963	0.0399	0.9770	0.9513

Table C.4 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap113-114, Cap121-124)

Problem	Cap113	Cap114	Cap121	Cap122	Cap123	Cap124
Size	50*50	50*50	50*50	50*50	50*50	50*50
Optimum	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R1	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R2	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R3	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R4	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R5	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R6	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R7	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R8	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R9	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R10	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R11	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R12	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R13	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R14	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R15	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R16	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R17	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R18	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R19	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R20	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R21	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R22	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R23	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R24	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R25	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R26	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R27	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R28	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R29	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R30	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Best Fitness	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Worse Fitness	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	1.3651	0.3635	1.4276	1.0240	1.3651	0.3635
Minimum CPU	0.2810	0.0470	0.0630	0.1090	0.2820	0.0470
Maximum CPU	3.4060	1.3600	3.5000	4.9380	3.4060	1.3590
Std Dev. of CPU	0.9785	0.3504	0.9811	0.9506	0.9781	0.3505

Table C.5 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (Cap131-134)

Problem	Cap131	Cap132	Cap133	Cap134
Size	50×50	50×50	50×50	50×50
Optimum	793439.56	851495.33	893076.71	928941.75
R1	793439.56	851495.33	893076.71	928941.75
R2	793439.56	851495.33	893076.71	928941.75
R3	793439.56	851495.33	893076.71	928941.75
R4	793439.56	851495.33	893076.71	928941.75
R5	793439.56	851495.33	893076.71	928941.75
R6	793439.56	851495.33	893076.71	928941.75
R7	793439.56	851495.33	893076.71	928941.75
R8	793439.56	851495.33	893076.71	928941.75
R9	793439.56	851495.33	893076.71	928941.75
R10	793439.56	851495.33	893076.71	928941.75
R11	793439.56	851495.33	893076.71	928941.75
R12	793439.56	851495.33	893076.71	928941.75
R13	793439.56	851495.33	893076.71	928941.75
R14	793439.56	851495.33	893076.71	928941.75
R15	793439.56	851495.33	893076.71	928941.75
R16	793439.56	851495.33	893076.71	928941.75
R17	793439.56	851495.33	893076.71	928941.75
R18	793439.56	851495.33	893076.71	928941.75
R19	793439.56	851495.33	893076.71	928941.75
R20	793439.56	851495.33	893076.71	928941.75
R21	793439.56	851495.33	893076.71	928941.75
R22	793439.56	851495.33	893076.71	928941.75
R23	793439.56	851495.33	893076.71	928941.75
R24	793439.56	851495.33	893076.71	928941.75
R25	793439.56	851495.33	893076.71	928941.75
R26	793439.56	851495.33	893076.71	928941.75
R27	793439.56	851495.33	893076.71	928941.75
R28	793439.56	851495.33	893076.71	928941.75
R29	793439.56	851495.33	893076.71	928941.75
R30	793439.56	851495.33	893076.71	928941.75
Best Fitness	793439.56	851495.33	893076.71	928941.75
Worse Fitness	793439.56	851495.33	893076.71	928941.75
Std Dev. of Fitness	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00
Average CPU	1.4281	1.0245	1.3651	0.3635
Minimum CPU	0.0630	0.1090	0.2810	0.0470
Maximum CPU	3.5000	4.9380	3.3910	1.3600
Std Dev. of CPU	0.9794	0.9508	0.9780	0.3502

Table C.6 Experimental Results of Fitness and CPU for $CPSO_{LS}$ (CapA, CapB, CapC)

Problem	CapA	CapB	CapC
Size	100×1000	100×1000	100×1000
Optimum	17156454.48	12979071.58	11505594.33
R1	17156454.48	13057343.82	11509361.66
R2	17156454.48	12979071.58	11518684.91
R3	17156454.48	13057343.82	11509361.66
R4	17156454.48	13057343.82	11509361.66
R5	17156454.48	12979071.58	11509361.66
R6	17156454.48	12979071.58	11509361.66
R7	17156454.48	13057343.82	11509361.66
R8	17156454.48	13007092.03	11528362.06
R9	17156454.48	13060291.10	11509361.66
R10	17156454.48	12979071.58	11535255.51
R11	17156454.48	13007092.03	11509361.66
R12	17156454.48	13057343.82	11509361.66
R13	17156454.48	12979071.58	11509361.66
R14	17156454.48	12979071.58	11509361.66
R15	17156454.48	12979071.58	11554372.62
R16	17156454.48	12979071.58	11509361.66
R17	17156454.48	13057343.82	11580061.89
R18	17156454.48	13084984.12	11509361.66
R19	17156454.48	13057343.82	11509361.66
R20	17156454.48	12979071.58	11509361.66
R21	17346752.16	12991528.78	11509361.66
R22	17156454.48	13057343.82	11509361.66
R23	17156454.48	13057343.82	11516305.38
R24	17156454.48	13057343.82	11509361.66
R25	17156454.48	12979071.58	11509361.66
R26	17156454.48	13057343.82	11509361.66
R27	17156454.48	13057343.82	11509361.66
R28	17156454.48	13057343.82	11509361.66
R29	17156454.48	12979071.58	11535255.51
R30	17156454.48	12979071.58	11509361.66
Best Fitness	17156454.48	12979071.58	11509361.66
Worse Fitness	17346752.16	13084984.12	11580061.89
Std Dev. of Fitness	34743.44	39458.67	16072.22
ARPE	0.04	0.33	0.09
HR	0.97	0.40	0.00
Average CPU	22.3921	30.6541	27.4234
Minimum CPU	6.1969	3.5375	5.5250
Maximum CPU	56.4968	60.4703	57.5594
Std Dev. of CPU	13.4863	16.3912	13.5369

APPENDIX D

EXPERIMENTAL RESULTS FOR $DPSO_{LS}$

Table D.1 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap71-74, Cap81-82)

Problem	Cap71	Cap72	Cap73	Cap74	Cap81	Cap82
Size	16×50	16×50	16×50	16×50	25*50	25*50
Optimum	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R1	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R2	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R3	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R4	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R5	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R6	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R7	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R8	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R9	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R10	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R11	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R12	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R13	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R14	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R15	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R16	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R17	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R18	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R19	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R20	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R21	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R22	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R23	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R24	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R25	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R26	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R27	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R28	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R29	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
R30	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Best Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Worse Fitness	932615.75	977799.40	1010641.45	1034976.98	796648.44	854704.20
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.0130	0.0078	0.0203	0.0109	0.1516	0.0557
Minimum CPU	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Maximum CPU	0.0470	0.0310	0.1090	0.0310	0.5160	0.3430
Std Dev. of CPU	0.0160	0.0106	0.0232	0.0123	0.1618	0.0709

Table D.2 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap83-84, Cap91-94)

Problem	Cap83	Cap84	Cap91	Cap92	Cap93	Cap94
Size	25*50	25*50	25*50	25*50	25*50	25*50
Optimum	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R1	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R2	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R3	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R4	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R5	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R6	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R7	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R8	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R9	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R10	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R11	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R12	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R13	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R14	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R15	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R16	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R17	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R18	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R19	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R20	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R21	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R22	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R23	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R24	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R25	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R26	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R27	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R28	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R29	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
R30	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Best Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Worse Fitness	893782.11	928941.75	796648.44	854704.20	893782.11	928941.75
Std Dev. of Fitness	0.00	0.00	0.00	0.00	0.00	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	1.00	1.00
Average CPU	0.1693	0.0339	0.1500	0.0562	0.1693	0.0344
Minimum CPU	0.0160	0.0000	0.0000	0.0000	0.0160	0.0000
Maximum CPU	0.5780	0.2190	0.5150	0.3590	0.5780	0.2190
Std Dev. of CPU	0.1513	0.0486	0.1624	0.0740	0.1532	0.0475

Table D.3 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap101-104, Cap111-112)

Problem	Cap101	Cap102	Cap103	Cap104	Cap111	Cap112
Size	25×50	25×50	25×50	25×50	50*50	50*50
Optimum	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R1	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R2	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R3	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R4	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R5	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R6	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R7	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R8	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R9	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R10	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R11	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R12	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R13	796648.44	854704.20	893782.11	928941.75	794299.85	851495.33
R14	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R15	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R16	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R17	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R18	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R19	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R20	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R21	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R22	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R23	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R24	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R25	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R26	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R27	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R28	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R29	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
R30	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Best Fitness	796648.44	854704.20	893782.11	928941.75	793439.56	851495.33
Worse Fitness	796648.44	854704.20	893782.11	928941.75	794299.85	851495.33
Std Dev. of Fitness	0.00	0.00	0.00	0.00	157.07	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	1.00	1.00	1.00	1.00	0.97	1.00
Average CPU	0.1505	0.0557	0.1693	0.0344	0.9927	0.7750
Minimum CPU	0.0000	0.0000	0.0150	0.0000	0.0620	0.0790
Maximum CPU	0.5320	0.3440	0.5780	0.2190	2.9530	1.6560
Std Dev. of CPU	0.1621	0.0712	0.1512	0.0487	0.7057	0.4639

Table D.4 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap113-114, Cap121-124)

Problem	Cap113	Cap114	Cap121	Cap122	Cap123	Cap124
Size	50*50	50*50	50*50	50*50	50*50	50*50
Optimum	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R1	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R2	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R3	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R4	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R5	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R6	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R7	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R8	893251.51	928941.75	793439.56	851495.33	893251.51	928941.75
R9	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R10	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R11	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R12	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R13	893076.71	928941.75	794299.85	851495.33	893076.71	928941.75
R14	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R15	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R16	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R17	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R18	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R19	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R20	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R21	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R22	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R23	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R24	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R25	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R26	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R27	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R28	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R29	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
R30	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Best Fitness	893076.71	928941.75	793439.56	851495.33	893076.71	928941.75
Worse Fitness	893251.51	928941.75	794299.85	851495.33	893251.51	928941.75
Std Dev. of Fitness	31.91	0.00	157.07	0.00	31.91	0.00
ARPE	0.00	0.00	0.00	0.00	0.00	0.00
HR	0.97	1.00	0.97	1.00	0.97	1.00
Average CPU	1.0510	0.5594	0.9932	0.7740	1.0510	0.5594
Minimum CPU	0.0160	0.0160	0.0630	0.0790	0.0160	0.0160
Maximum CPU	3.0160	1.8280	2.9530	1.6410	3.0160	1.8280
Std Dev. of CPU	0.8621	0.5622	0.7059	0.4625	0.8615	0.5625

Table D.5 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (Cap131-134)

Problem	Cap131	Cap132	Cap133	Cap134
Size	50×50	50×50	50×50	50×50
Optimum	793439.56	851495.33	893076.71	928941.75
R1	793439.56	851495.33	893076.71	928941.75
R2	793439.56	851495.33	893076.71	928941.75
R3	793439.56	851495.33	893076.71	928941.75
R4	793439.56	851495.33	893076.71	928941.75
R5	793439.56	851495.33	893076.71	928941.75
R6	793439.56	851495.33	893076.71	928941.75
R7	793439.56	851495.33	893076.71	928941.75
R8	793439.56	851495.33	893251.51	928941.75
R9	793439.56	851495.33	893076.71	928941.75
R10	793439.56	851495.33	893076.71	928941.75
R11	793439.56	851495.33	893076.71	928941.75
R12	793439.56	851495.33	893076.71	928941.75
R13	794299.85	851495.33	893076.71	928941.75
R14	793439.56	851495.33	893076.71	928941.75
R15	793439.56	851495.33	893076.71	928941.75
R16	793439.56	851495.33	893076.71	928941.75
R17	793439.56	851495.33	893076.71	928941.75
R18	793439.56	851495.33	893076.71	928941.75
R19	793439.56	851495.33	893076.71	928941.75
R20	793439.56	851495.33	893076.71	928941.75
R21	793439.56	851495.33	893076.71	928941.75
R22	793439.56	851495.33	893076.71	928941.75
R23	793439.56	851495.33	893076.71	928941.75
R24	793439.56	851495.33	893076.71	928941.75
R25	793439.56	851495.33	893076.71	928941.75
R26	793439.56	851495.33	893076.71	928941.75
R27	793439.56	851495.33	893076.71	928941.75
R28	793439.56	851495.33	893076.71	928941.75
R29	793439.56	851495.33	893076.71	928941.75
R30	793439.56	851495.33	893076.71	928941.75
Best Fitness	793439.56	851495.33	893076.71	928941.75
Worse Fitness	794299.85	851495.33	893251.51	928941.75
Std Dev. of Fitness	157.07	0.00	31.91	0.00
ARPE	0.00	0.00	0.00	0.00
HR	0.97	1.00	0.97	1.00
Average CPU	0.9922	0.7745	1.0516	0.5594
Minimum CPU	0.0620	0.0780	0.0150	0.0160
Maximum CPU	2.9530	1.6400	3.0310	1.8280
Std Dev. of CPU	0.7060	0.4615	0.8639	0.5597

Table D.6 Experimental Results of Fitness and CPU for $DPSO_{LS}$ (CapA, CapB, CapC)

Problem	CapA	CapB	CapC
Size	100×1000	100×1000	100×1000
Optimum	17156454.48	12979071.58	11505594.33
R1	17426742.86	13060291.10	11535255.51
R2	17580326.65	13057343.82	11633180.53
R3	17413325.08	13084984.12	11509361.66
R4	17706093.45	12979071.58	11595517.65
R5	17346752.16	13103172.56	11553916.02
R6	17156454.48	12979071.58	11539375.97
R7	17689516.71	13057343.82	11528362.06
R8	17156454.48	13057343.82	11505594.33
R9	17346752.16	13060291.10	11540878.06
R10	17779775.71	13150805.13	11518684.91
R11	17346752.16	13179479.33	11518684.91
R12	17844165.28	13138281.80	11543204.97
R13	17180539.56	12979071.58	11543204.97
R14	17156454.48	13148369.52	11505594.33
R15	17426742.86	13098385.38	11528362.06
R16	17779775.71	12979071.58	11509361.66
R17	17156454.48	13214718.16	11630532.09
R18	17156454.48	12979071.58	11544611.67
R19	17156454.48	13444180.76	11560840.92
R20	17156454.48	13259918.72	11528362.06
R21	17771742.16	13070745.09	11716349.47
R22	17580326.65	13057343.82	11505594.33
R23	17156454.48	13103172.56	11595517.65
R24	17711146.02	13070745.09	11528362.06
R25	17156454.48	13193415.97	11509361.66
R26	17516002.50	13113656.24	11633619.04
R27	17665889.11	12979071.58	11509361.66
R28	17273028.38	13057343.82	11530826.61
R29	17535907.42	12991528.78	11505594.33
R30	17156454.48	13070745.09	11535255.51
Best Fitness	17156454.48	12979071.58	11505594.33
Worse Fitness	17844165.28	13444180.76	11716349.47
Std Dev. of Fitness	243932.83	100126.81	49717.56
ARPE	1.51	0.86	0.37
HR	0.33	0.20	0.13
Average CPU	19.5889	19.6360	18.7685
Minimum CPU	6.5687	8.0343	6.3703
Maximum CPU	26.0703	23.7907	21.0016
Std Dev. of CPU	5.1268	4.5195	3.1724

APPENDIX E

EXAMPLE OF A RUN'S EXPORTED DATA

```
Problem Name....=cap122

Replication Number........=1
Seed=81569 Iteration=1000 Popsize=50 dimension=0
CPU in seconds............=3.6720
Relative Percent Deviation=0.8105
Optimal Value.............=851495.3310
Global Fitness............=858396.9250

      0     0     0     0     0     1     0     0     0     0     1
      0     0     0     1     0     0     1     0     0     0     0
      1     0     0     0     1     0     0     0     0     0     0
      1     0     0     0     1     1     0     0     0     0     0
      1     0     0     0     1     0

Problem Name....=cap122

Replication Number........=2
Seed=81570 Iteration=1000 Popsize=50 dimension=0
CPU in seconds............=3.6720
Relative Percent Deviation=0.1489
Optimal Value.............=851495.3310
Global Fitness............=852762.8750

      0     0     0     0     0     1     0     0     0     0     1
      0     1     0     1     0     0     0     0     0     0     0
      1     0     1     0     1     0     0     0     0     0     0
      1     0     0     1     0     0     0     0     0     0     0
      1     1     0     0     0     0

Problem Name....=cap122

Replication Number........=3
Seed=81571 Iteration=1000 Popsize=50 dimension=0
CPU in seconds............=3.5630
Relative Percent Deviation=0.0976
Optimal Value.............=851495.3310
Global Fitness............=852326.3875

      0     0     0     1     0     1     0     0     0     0     1
      0     1     0     1     0     0     0     0     0     0     0
      1     0     1     0     1     0     0     0     0     0     0
      1     0     0     0     0     0     0     0     0     0     0
      1     0     1     0     0     0
```

Figure E.1 Example of a run's exported data
(Cap122 problem's 1st, 2nd and 3rd replications results obtained with CPSO algorithm)

APPENDIX F

EXAMPLE OF A DATASET

Table F.1 Structure of a Dataset

	Facility: **1**	Facility: **2**	**…**	Facility: **n**
C: Capacity	C_1	C_2	..	C_n
F: Fixed Cost	fc_1	fc_2	..	fc_n
Customer: **1**	c_{11}	c_{21}	..	c_{n1}
Customer: **2**	c_{12}	c_{22}	..	c_{n2}
..	..	..	..	..
..	..	..	..	..
Customer: **m**	c_{1m}	c_{2m}		c_{nm}

The structure of a dataset in the OR-library is given in the Table F.1. Since these datasets are also use in capacitated facility location problem the capacities of facilities are given. As an example dataset of problem Cap71 is given in the Tables F.2 and F.3.

Table F.2 Dataset of problem Cap71 (a)

	1	2	3	4	5	6	7	8
C:	58268	58268	58268	58268	58268	58268	58268	58268
F:	7500	7500	7500	7500	7500	7500	7500	7500
1	6739.725	10355.05	7650.4	5219.5	5776.125	6641.175	4374.525	3847.1
2	3204.863	5457.075	3845.4	2396.85	2628.488	3220.088	1838.963	2266.35
3	4914	26409.6	19622.4	13876.8	9147.6	14977.2	21848.4	35330.4
4	32372.11	29982.23	21024.33	29681.4	21275.01	20071.71	64292.99	80186.58
5	1715.463	2152.175	1577.9	1061.75	1250.463	1363.613	1524.038	955.575
6	6421.513	23701.6	16197.03	10383.43	7483.613	12332.94	15840.66	27251.25
7	81972.38	28499.25	43134	65767.5	58805.63	48555.38	138615.4	155294.3
8	33391.46	26544.38	6370.65	16770.6	13571.66	8861.738	51550.54	57907.58
9	2020.838	2480.775	1869.45	1324.95	1525.838	1646.288	1817.063	1211.925
10	1459.6	1995.2	1402.4	869.6	1050.8	1181.2	1133.2	546.4
11	141015.4	205925.1	104130.3	12638.5	46089.31	66146.06	198300.8	220212.1
12	17684.5	32069.4	15322.8	8429.8	1231.7	9073.9	32781.3	41335.4
13	38207.63	42477.35	15319.7	15832.8	11526.43	5185.975	62653.18	71210.95
14	1953.738	5044.325	4089.8	3428.425	2289.788	3530.313	5553.763	8308.3
15	17181.56	36054.38	25399.5	16297.5	15828.56	21148.31	7310.813	21709.5
16	25640.85	35602.5	25154.4	15763.8	18421.65	21255.75	18478.05	8135.7
17	7031.425	10492.05	6305.4	2542.5	3918.275	4743.175	6856.275	7119
18	78453.7	92515.8	36644.4	27445.6	23562.5	23034.7	126332.7	141375
19	9452.713	12441.28	7754.45	3542	5082.138	6005.588	10274.96	10214.88
20	8597.063	14113.13	10500.75	7254	7875.563	9152.813	5467.313	6371.625
21	1581.275	2030.15	1326.2	693.5	924.825	1063.525	1628.775	1619.75
22	23170.99	48702.45	36072.9	26166.98	23493.79	30494.51	14919.41	33813.3
23	12087.56	19877.33	9670.05	3801.9	2252.213	5847.488	19650.04	23844.53
24	4883	12851.6	10822.4	8930	6798.2	9435.4	11943.4	18148.8
25	24063.88	39682.5	24603.15	11050.05	13644.68	18976.38	20197.38	24684.55
26	4124.038	12148.85	8180.675	5611.05	2952.963	5851.163	11613.86	17111.18
27	281463	406770	325852.8	253234.8	264755.4	294457.8	211356.6	210756
28	11056.76	22113.53	11424.6	5582.475	2430.613	7436.088	19279.01	25460.13
29	8585.625	22449.15	14122.6	7458.95	6609.425	10790.78	11525.83	20448.85
30	12480.19	25455.38	22151.25	19069.88	15598.69	19892.81	23976.56	34080.75
31	3727.763	11116.88	8229.375	5826.975	4628.663	6632.588	6476.663	11884.95
32	4673.025	13346.9	7880.95	4330.9	2861.775	5655.125	9623.775	14610.75
33	13451.69	35106.25	25927.25	17347.63	15249.81	21192.19	16808.19	32845.75
34	372672.6	229188	203364	322800	261306.6	229995	681269.4	810550.8
35	9745.938	18070	12049.38	7198.75	7592.813	9802.813	5780.938	10692.5
36	12055.13	18181.05	11400.9	5307	7379.475	8870.925	10865.63	10202.25
37	97602.71	73603.55	59561.98	83331.7	65940.34	56946.39	185247.8	222003.7
38	60774.51	63568.43	27330.55	30982	20497.91	15076.06	97731.61	114578.1
39	54470.06	65177.25	52117.13	40378.88	44494.31	47243.81	47631.56	34351.13
40	7146.3	8618.2	6428.8	7822.8	5211.1	5621.1	15256.1	19762
41	38011.61	70728.08	39587.55	15801.4	16494.81	27925.61	49862.66	62659.28
42	39723.31	52917.88	32225.45	13627.4	20427.14	24504.19	41119.56	40854.28
43	16111.56	20714.38	15620	11041.25	12598.44	13719.06	12502.19	7885.625
44	16981.25	32575	23312.5	16250	16268.75	19856.25	9331.25	20750
45	168663.3	210766.1	169251.5	131938.9	144628.5	153760.6	134768.1	109304.8
46	57109.86	66703	53124.18	40919.73	45381.86	48057.31	52583.59	39050.58
47	15576.08	18481.5	14368.95	10672.65	12024.08	12834.38	14205.23	10134.3
48	2542.488	3928.575	3020.85	2205	2361.188	2682.138	1756.038	1983.275
49	34056.3	34221	24448.8	31329.6	21905.1	20807.1	69009.3	86632.2
50	7095.675	11999.1	7886.55	4190.25	4847.925	6351.975	4903.425	6421.35

Table F.3 Dataset of problem Cap71 (b)

	9	10	11	12	13	14	15	16
C:	58268	58268	58268	58268	58268	58268	58268	58268
F:	7500	7500	0	7500	7500	7500	7500	7500
1	6429.475	5396.525	5219.5	4182.9	7391.25	5038.825	10349.58	6051.7
2	3117.863	2582.813	2296.8	1779.15	5115.6	2189.138	5399.438	2838.375
3	15111.6	23679.6	9828	19303.2	57472.8	11180.4	22957.2	15489.6
4	25921.09	69206.46	23096.68	48700.23	135170.7	40527.81	60515.96	52911.78
5	1318.663	1789.088	1133.05	1015.25	2005.7	1379.888	2512.938	1823.575
6	12444.74	17769.21	7029.425	13919.1	45474.65	6966.538	20326.64	10956.4
7	53176.88	147325.1	56998.5	102384	259515	96429.38	131920.1	118381.5
8	10985.29	57376.69	12741.3	33595.65	105796.4	32220.79	60071.96	46527.53
9	1593.488	2099.213	1395.9	1275.45	1940.4	1663.613	2869.763	2135.925
10	1134.8	1406.8	928.8	768.8	1950.4	1145.2	2314.8	1500.8
11	58178.31	241573.9	25277	97536.25	461992.1	106122.2	288418.8	185456.3
12	9254.7	37595.1	2463.4	19775	79235.6	16712.7	41956.9	28589
13	11563.08	70496.28	10371.95	38482.5	135275.2	36631.68	77569.73	55891.25
14	3558.913	5947.013	2434.575	4897.75	13134.55	3283.638	4634.988	4204.2
15	20779.31	12569.06	14621.63	11931	39913.5	11169.94	32479.69	14375.63
16	20437.95	23300.25	16271.4	12619.5	35094.9	19253.55	39867.75	24957
17	4415.475	8788.575	3062.3	2712	17458.5	5031.325	13093.88	8294.2
18	11423.1	144881.1	22846.2	74042.8	274079	75211.5	159433.3	114834.2
19	5638.738	12438.11	4123.9	5635.575	21789.63	8073.863	16239.44	11726.55
20	8870.063	6729.938	7312.5	5869.5	9506.25	6695.813	13554.94	7707.375
21	1008.425	1953.675	780.9	931.95	3066.6	1298.175	2600.625	1846.8
22	30655.91	11166.86	22333.73	20982	56025.98	17380.76	40178.51	15978.6
23	6260.738	22873.39	0	10703.18	46945.2	10145.29	26881.91	17728.43
24	9496.2	11616.6	7106	10898.4	28226.4	7117.4	7725.4	7911.6
25	17796.08	27157.08	10541.3	5270.65	59259.2	14845.33	44597.03	26495.7
26	5918.563	12852.34	3268.9	9073.725	29479.08	5623.688	13172.49	8745.15
27	289434.6	239639.4	248102.4	222222	124051.2	238875	392519.4	261534
28	7551.488	22351.54	1601.175	11698.68	49650.85	9022.838	26549.21	16963.8
29	10887.18	14092.48	5916.55	8953.15	37089.9	2958.275	20575.38	9917.15
30	19991.81	23444.44	16099.88	22275	50490	16118.44	8049.938	17411.63
31	6678.788	6101.288	4440.975	6843.375	18722.55	3693.113	8596.088	3285.975
32	5719.525	11338.43	2398.9	6931.05	26701.85	3900.225	13310.68	7961.45
33	21329.19	15695.06	14693.25	19385.5	53121.75	10951.44	27888.06	7346.625
34	258078.6	728398.2	261790.8	512606.4	1361570	451435.8	644793	575875.2
35	9607.813	8559.688	6353.75	4891.25	21336.25	5951.563	17830.31	9514.375
36	8340.225	13994.93	5984.1	4154.1	26946.75	8459.175	21470.48	13697.55
37	73007.01	198738.8	65986.23	137295.4	378755.4	119995.8	174969	155375.1
38	24757.94	109515.8	20525.58	63513.1	209073.2	58395.54	114439.8	87468.83
39	46820.81	53659.31	42775.88	39250.88	40960.5	48318.94	73399.31	55712.63
40	7047.9	16305.7	5658	11939.2	32644.2	9417.7	13583.3	12308.2
41	26917.01	61965.86	10086	22567.43	133135.2	23134.76	81255.34	47404.2
42	22884.54	50669.91	16196.5	20636.58	91957.03	31401.66	69686.84	47528.35
43	13320.31	14853.44	11550	10071.88	13193.13	13052.19	23103.44	15654.38
44	19556.25	9543.75	14550	13087.5	32087.5	12481.25	27518.75	12525
45	150511.2	149278.6	136084.7	124879.7	89976.15	142499.6	224408.1	160511.6
46	46994.46	58814.09	42605.63	40553.23	45922.45	49175.14	75966.29	61003.93
47	12512.48	16103.33	11183.25	10561.65	14940.6	13172.93	21287.03	16755.45
48	2611.088	2073.313	2174.375	1857.1	2006.55	2064.738	3788.313	2318.925
49	27212.1	74352.9	23899.8	51935.4	147937.2	42986.7	64873.5	56510.4
50	6030.075	6801.525	4001.55	2614.05	14979.45	4503.825	12617.93	7448.1